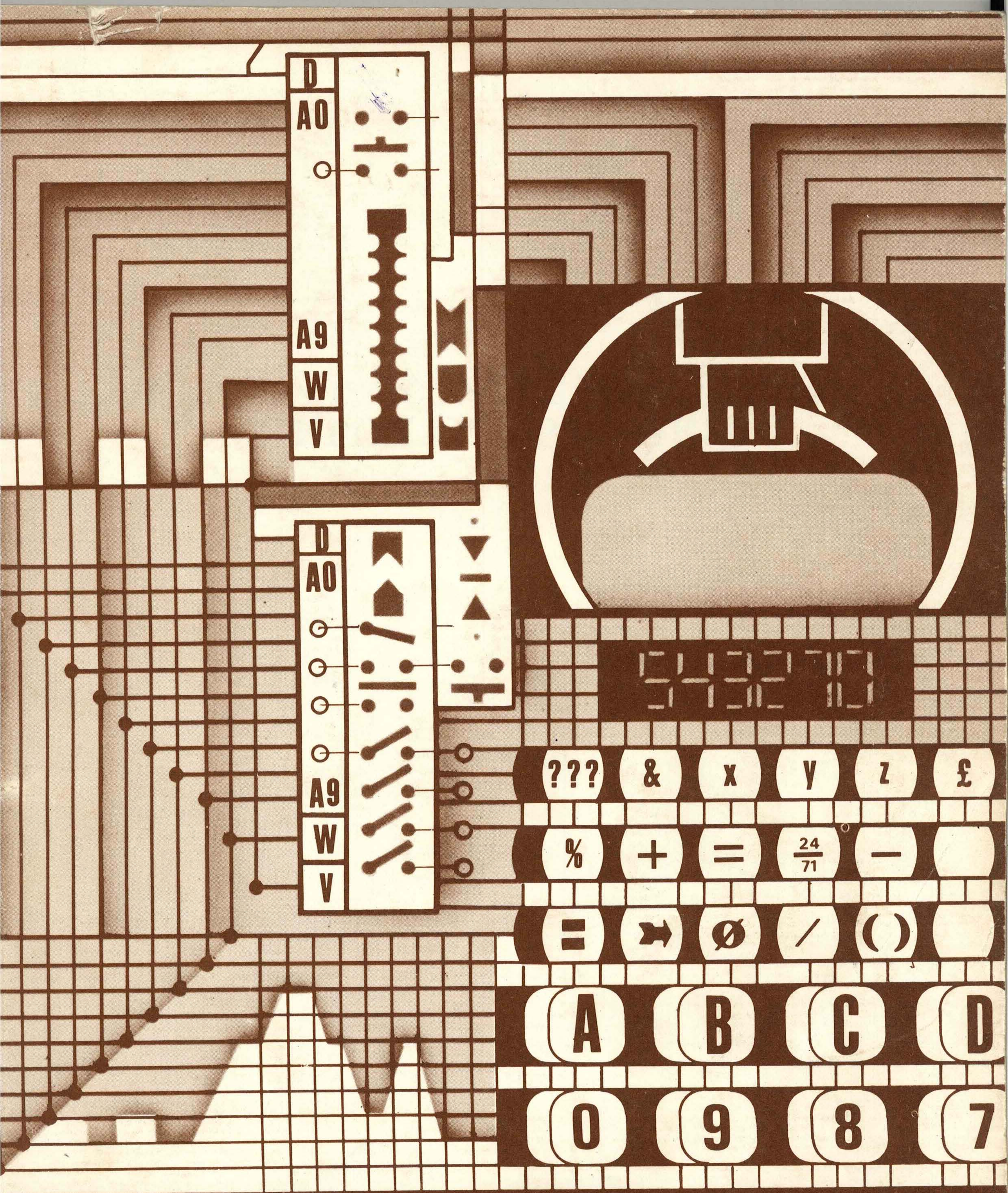


МИКРОПРОЦЕССОРНАЯ ЛАБОРАТОРИЯ «МИКРОЛАБ КР580ИК80»

МИКРОПРОЦЕССОРНАЯ ЛАБОРАТОРИЯ «МИКРОЛАБ КР580ИК80»

РЕКОМЕНДАЦИИ ПО РАЗРАБОТКЕ АППАРАТНО-ПРОГРАММНЫХ СРЕДСТВ ДЛЯ УСТРОЙСТВ НА БАЗЕ МИКРОПРОЦЕССОРА КР580ИК80

Настоящий курс предназначен для изучения элементов аппаратного и программного обеспечения систем, построенных на базе микропроцессора КР580ИК80.

В первом разделе даны общие принципы построения устройств с использованием микропроцессора. Описывается в общих чертах функционирование микропроцессора, его связь с блоком памяти и с периферийными устройствами. Даются основы программирования микропроцессорных систем.

Во втором разделе на примере "Микролаб КР580ИК80" рассматриваются аппаратные средства микро-ЭВМ, принципы построения отдельных узлов. Описывается взаимодействие микропроцессора с оперативным запоминающим устройством, с постоянным запоминающим устройством, с клавиатурой /через устройство ввода/вывода/, рассматривается организация вывода данных на индикацию.

В третьем разделе рассматривается набор команд микропроцессора КР580ИК80. Даны примеры написания программ и исполнения их на микро-ЭВМ "Микролаб КР580ИК80".

В четвертом разделе приводится методика поиска неисправностей в микропроцессорной системе. Описывается контрольно-измерительная аппаратура и ее использование при диагностике микропроцессорных устройств.

Таким образом, изучая настоящий курс, специалисты получат рекомендации по разработке аппаратных и программных средств микропроцессорных систем, что облегчит разработку устройств со встроенным микропроцессором.

ВВЕДЕНИЕ. МИКРОПРОЦЕССОРЫ И МИКРО-ЭВМ

Урок 1

ОСНОВНЫЕ ЭЛЕМЕНТЫ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

Микропроцессор /МП/ - устройство, выполняющее функции программной обработки информации, аналогичные функциям центрального процессора обычной ЭВМ, и реализованное на одной схеме большой степени интеграции /БИС/ или в виде модуля, содержащего несколько БИС.

Ранние ЭВМ выполнялись с применением тысяч электронных ламп. Они были громоздкими и неудобными. Следующее поколение ЭВМ строилось на транзисторах, что сделало их значительно надежнее, уменьшило их размеры и стоимость. Это положило начало практическому использованию компьютера. В 1960 г. были изготовлены маленькие и мощные ЭВМ на основе сотен вентилей, триггеров и других подобных им интегральных схем - схемы малой степени интеграции /МИС/. По мере развития полупроводниковой технологии стало возможным совместить в одной интеграль-

ной схеме несколько десятков вентилей - микросхем средней степени интеграции /СИС/ - /счетчик,дешифраторы, регистры и сумматоры/.

Работы по миниатюризации ЭВМ продолжались, и в 1971 г. был создан первый МП. Микропроцессорный кристалл содержит тысячи вентилей и является интегральной схемой БИС.

Кроме того,получили развитие БИС памяти, хранящие тысячи бит цифровой информации в одном корпусе. Использование БИС дало возможность уменьшить размеры и снизить стоимость небольших ЭВМ. Применение МП позволило начать производство миниатюрных недорогих компьютеров. Сейчас МП применяется во многих случаях там, где раньше использовали схемную логику. Микропроцессорные устройства гораздо дешевле и имеют меньше компонентов, чем их предшественники на основе МИС и СИС. Небольшие микропроцессорные системы можно строить на нескольких микросхемах. Часто это заменяет платы с несколькими десятками микросхем с меньшей степенью интеграции. Вследствие значительного уменьшения числа дискретных компонентов и внутренних связей значительно увеличилась надежность. Уменьшение числа микросхем МИС и СИС также возможно за счет использования специализированных интегральных схем вместо МП. Однако разработка специализированных интегральных схем - очень сложный и дорогой процесс и может быть оправдана только при производстве больших серий. МП позволяет использовать для достижения миниатюризации стандартные микросхемы. Алгоритм работы определяется программой, которая хранится в памяти. Производство микросхем памяти с регулярной топологией относительно недорого. Микропроцессорные системы могут осуществлять самодиагностику. Такие приборы,как цифровые вольтметры, могут выполнять функции автоматического усреднения по числу измерений, сложения или вычитания величин при каждом измерении, самокалибровку и др. Применение приборов со встроенным МП позволяет использовать клавиатуру вместо многопозиционных переключателей на передней панели и обеспечивает возможность дистанционного управления.

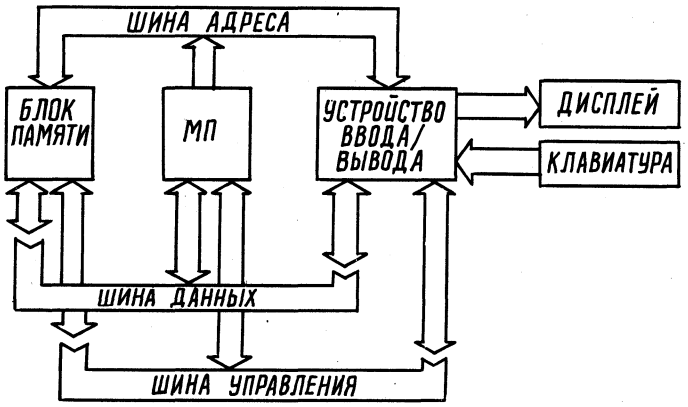


Рис.1. Схема микропроцессорной системы

На рис.1. показана упрощенная схема микропроцессорной системы.

Микро-ЭВМ "Микролаб КР580ИК80" /рис.2 /, предназначенная для изучения аппаратной части и программного обеспечения, имеет ту же структуру.

МП /называемый также процессором/ является "мозгом" системы. Он содержит всю логику для распознавания и исполнения списка команд /программы/. Память хранит программу и данные. Чтобы обратиться к

Рис.2. Микролаб КР580ИК80

программе, хранимой в памяти "Микролаб КР580ИК80", необходимо:

1.Открыть микро-ЭВМ и включить питание, тумблер режима поставить в положение АВТ.

2.Нажать кнопку СБРОС - на индикаторах загорятся все нули,

3.Нажать кнопки 3, 0, 0 - число 0300 высветится на правых индикаторах.

4.Нажать кнопку УСТ.АД. - число 0300 высветится на левых индикаторах, а на двух крайних правых индикаторах будет содержимое ячейки памяти с адресом 0300.

5.Нажать кнопку ПУСК - начнется выполнение программы, записанной в памяти с адреса 0300.

Устройство ввода/вывода связывает МП с клавиатурой. Передача информации внутри системы осуществляется по трем шинам: шине адреса, шине данных и шине управления. Под шиной следует понимать группу проводников, связывающих все устройства системы, передающих однотипные сигналы. Количество проводников в шине определяется разрядностью передаваемой по ней информации. Проводники, соответствующие одноименным разрядам всех устройств системы, в каждой шине соединены параллельно. МП использует адресную шину для выбора ячейки памяти или устройства ввода/вывода. Выбрав информацию по данному адресу с помощью адресной шины, МП пересылает ее по шине данных. Информация может передаваться из МП в память и устройство ввода/вывода или из устройства ввода/вывода и памяти в МП. Заметим, что МП, как правило, участвует во всех операциях переноса данных. Управляющая шина предназначена для передачи управляющих сигналов, признаков состояния периферийного оборудования и процессора. Один знак двоичной информации /1,0/ называется битом. Электрический бит информации - это логический уровень напряжения сигнала. Амплитуда логических уровней дискретна: лог. 1 - высокий уровень, лог. 0 - низкий. МП обращаются с данными не как с отдельными битами, а как с группами битов, называемых "словами". Наиболее распространены МП, оперирующие 8-битными словами или байтами. Эти МП называются 8-битными. Для 8-битных процессоров байт и "слово" синонимы, однако термин "слово" используется также для обозначения группы из 16 и более бит.

ПРОГРАММЫ

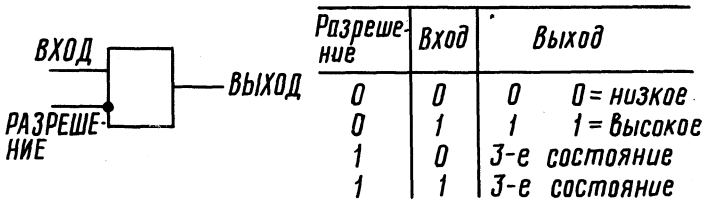
Для выполнения системой необходимой задачи требуется соответствующий список команд:

1/ считать данные с клавиатуры;
2/ записать данные на дисплей;
3/ повторить с шагом 1.

Чтобы МП мог выполнить задачу в соответствии со списком команд, команды должны быть переведены в код, понятный МП. Эти коды затем сохраняются в памяти системы. МП начинает работу, считывая первый код из памяти, расшифровывает соответствующую команду и выполняет указанную в ней операцию. Затем процессор считывает команду из следующей ячейки памяти и снова выполняет соответствующую операцию. Этот процесс повторяется с последующим перебором соответствующих ячеек. Определенные команды заставляют МП "перепрыгнуть" на другой адрес памяти, минуя установленный порядок очередности, для выполнения последующей команды. Программа может возвратить МП к выполнению команды с адресом в начале программы, образуя цикл, который повторно выполняется. Это позволяет создать короткую программу для операций, повторяющихся много раз.

ДРАЙВЕРЫ С ТРЕМЯ УСТОЙЧИВЫМИ СОСТОЯНИЯМИ

Все устройства в микропроцессорной системе обмениваются информацией посредством одной и той же шины данных. МП выбирает какое-то одно устройство для вывода его данных на шину данных и отключает другие. Такой режим работы требует, чтобы устройства имели три устойчивых состояния, которые позволяли бы процессору обмениваться информацией в каждый момент времени только с одним выбранным устройством. На рис.3. показан тристабильный драйвер и его таблица истинности. Буфер имеет разрешающий вход. Когда



Разрешение	Вход	Выход
0	0	0 = низкое
0	1	1 = Высокое
1	0	3-е состояние
1	1	3-е состояние

Рис.3. Тристабильный драйвер

уровень напряжения на разрешающем входе низкий, буфер работает как обычно, т.е. сигнал со входа подается на выход. Если же на разрешающем входе высокий уровень, выходной сигнал имеет третье состояние, которое называют состоянием с высоким импедансом.

На рис.4. показана эквивалентная цепь, которая обеспечивает третье состояние.

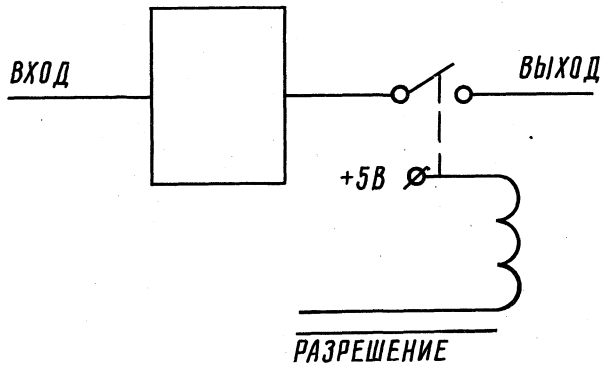


Рис.4. Эквивалентная схема тристабильного драйвера

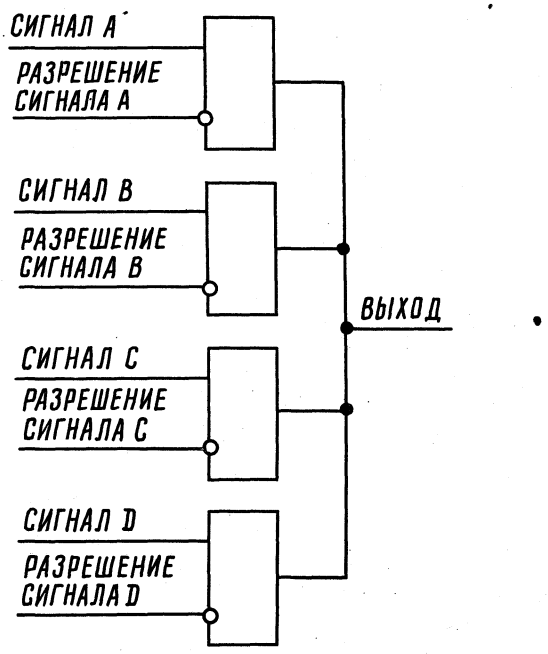


Рис.5. Выход четырех сигналов на общую линию

Тристабильные драйверы позволяют совместно работать нескольким устройствам при использовании одной линии данных. На рис.5. показана цепь, на которую поступают четыре различных сигнала. В случае одновременного пропускания их на выход, логическое состояние схемы было бы неопределенным. Поэтому необходимо использовать сигналы управления, которые бы разделяли во времени поступающую входную информацию. Для этого были разработаны интегральные схемы, содержащие тристабильные драйверы. Так например, МП и память имеют специальный вывод, по которому можно управлять их выходными драйверами.

МИКРОПРОЦЕССОР

На рис.6. показаны основные сигналы типичного МП. Здесь мы видим 16 адресных линий, образующих шину адреса, и восемь линий данных, образующих шину данных. Шина данных - двунаправленная. Это означает, что данные по ней могут проходить в обоих направлениях. Сигналы ЗАПИСЬ и ЧТЕНИЕ - сигналы, управляющие потоком данных по шине данных /т.е. определяют направление движения данных/. Сигналы СБРОС и ПРЕРЫВАНИЕ обеспечивают добавочные функции управления: сигнал СБРОС используется для приведения в исходное состояние внутренней схемы МП, сигнал Прерывание позволяет МП пре-

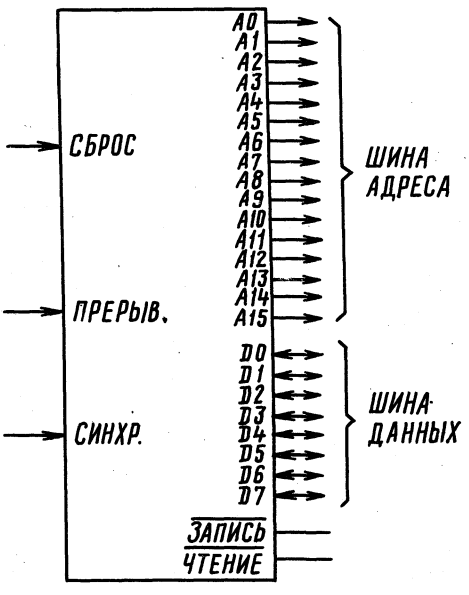


Рис.6. Основные сигналы МП

рывать выполнение текущей задачи для немедленного решения другой задачи. Использование всех сигналов будет описано в разделах, касающихся аппаратной части микропроцессорных систем. Сигнал СИНХР. синхронизирует все устройства в системе и устанавливает темп выполнения команд.

СТРУКТУРА ПАМЯТИ

Обычно в микропроцессорных системах используют интегральные схемы памяти для хранения программ и данных. Существуют микросхемы, способные хранить более 64 тыс. битов в одном кристалле. 65536-битная память может хранить свыше 8 тыс. символов или же около трех страниц машинописного текста на кристалле площадью около 1 $см^2$. Простейшим запоминающим устройством является триггер, который хранит 1 бит информации. Регистры содержат несколько триггеров в одном корпусе с общим входом синхронизации. Технология БИС позволяет располагать тысячи триггеров в одном корпусе, но однако при соединении тысяч триггеров в одном корпусе невозможно иметь отдельный вывод для каждого. Для решения этой проблемы используют адресные входы для выбора интересующего нас триггера. Дешифратор, расположенный в кристалле памяти, расшифровывает адреса и связывает выбранную ячейку с выводом данных. На рис.7. показана принципиальная схема 8-битной памяти /существуют гораздо более крупные объемы памяти/. Для простоты указаны только выходные цепи данных. Дешифратор преобразует двоичный адрес на входе в восемь отдельных выходов, один для каждой возможной комбинации на трех адресных линиях. Эти сигналы управляют тристабильными драйверами на выходе каждой ячейки памяти /триггера/. Данные из адресованной ячейки помещаются на линию вывода данных. Такой механизм позволяет использовать одну линию данных для всех ячеек памяти в кристалле. Структура памяти может содержать не только 1, но 4, и 8 разрядов в зависимости от типа интегральной схемы /ИС/. Если ИС имеет 8 информационных входов, то каждая

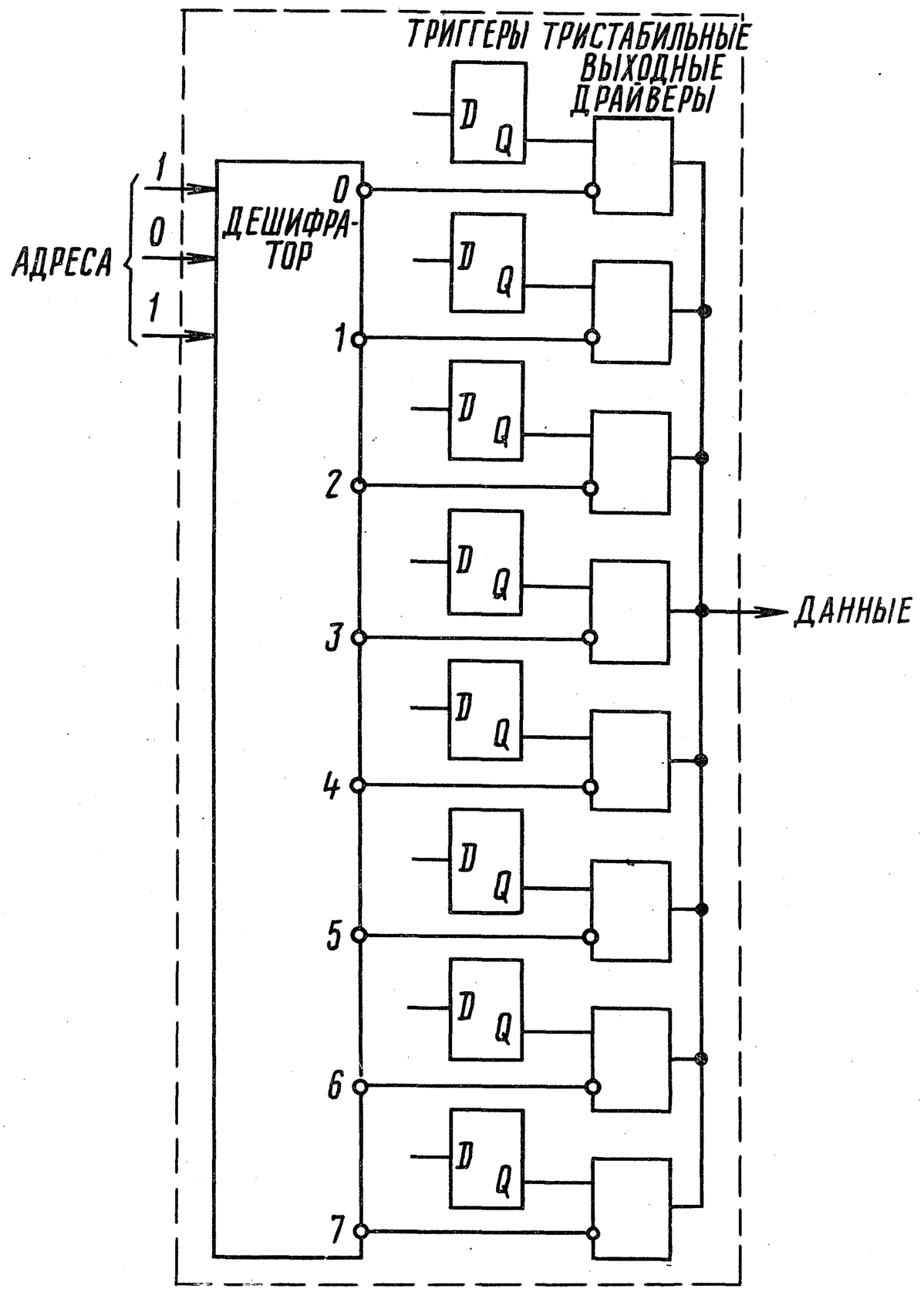


Рис.7. Принципиальная схема памяти на 8 бит

ячейка памяти хранит 8 битов информации. Несмотря на то, что память может содержать тысячи ячеек, одновременно может быть доступной только одна. Число адресуемых ячеек зависит от числа адресных линий. Если адресная линия одна, то можно выделить две ячейки с адресами: 0 или 1. Используя две адресные линии, можно выбрать одну из четырех ячеек: 00, 01, 10, 11. Обычно число ячеек 2^N, где N - число адресных линий.

ПОСТОЯННЫЕ И ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Интегральные схемы памяти, используемые в микропроцессорных системах, подразделяют на две категории: постоянные запоминающие устройства /ПЗУ/ и оперативные запоминающие устройства /ОЗУ/. ПЗУ -

это память, с которой можно только считывать. Информация запрограммирована в нее еще при изготовлении или же с помощью специальной процедуры - программирования до установки ее в схему. Программа, записываемая в ПЗУ, часто является программой стандартного матобеспечения. ОЗУ - это память, в которой данные могут храниться и изменяться. Важной особенностью полупроводниковых ОЗУ является их непостоянность: они теряют свою информацию при отключении питания и невозможно предсказать, какая информация будет записана в ОЗУ после его включения. Содержимое ПЗУ, наоборот,не может быть изменено,поэтому оно используется для хранения постоянной программы и постоянных данных. Для временных программ и хранения данных необходимо использовать ОЗУ. На рис.8 показано ПЗУ, содержащее 2048 слов по восемь разрядов каждое или 16348 битов. При использовании больших объемов памяти вводится число К=1024 /2^{10} бит/. Таким образом, вышеуказанная память содержит 2К байтов или 16К битов. Так как каждая ячейка содержит 8 битов, то такая структура описывается как 2Кх8 ПЗУ. Когда на входе ВК /выборка кристалла/ низкий уровень, выходные драйверы ПЗУ приведены в действие. Когда уровень на ВК высокий, информационные выходы находятся в состоянии с высоким импедансом. Тристабильные выходы позволяют объединить линии данных многих устройств и выбрать одно устройство путем подачи на вход ВК низкого уровня.

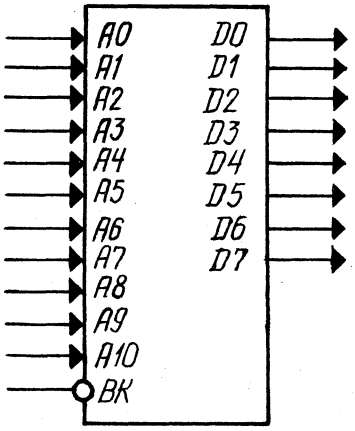


Рис.8. 2Кх8 ПЗУ

На рис.9 показано 1Кх8 ОЗУ.Это ОЗУ содержит 1024 восьмибитных ячейки. Линии данных являются двунаправленными, так как данные могут поступать в память или считываться из памяти. ОЗУ имеет добавочный управляющий вход, называемый ЗАПИСЬ/ЧТЕНИЕ. Для записи данных в ОЗУ выбирается адрес, данные поступают по линиям данных и на входе ЗАПИСЬ/ЧТЕНИЕ устанавливается низкий уровень. Если в этот момент на входе ВК также низкий уровень, то информация заносится в память.

Обычно линия записи активна при низком уровне на ней. Это означает, что если уровень высокий, то производится считывание, если низкий - запись. Заметим, что этот вход не активен до установления ВК.

ПЗУ и ОЗУ бывают самых различных типов и с разным количеством слов и разрядов. В последующих разделах ПЗУ и ОЗУ будут рассмотрены более подробно.

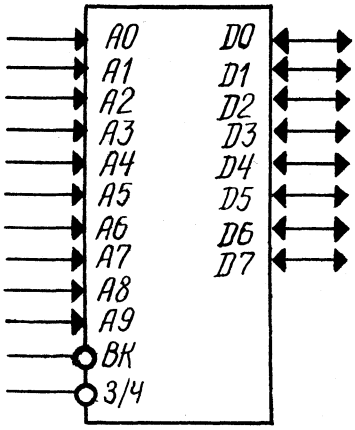


Рис.9. 1Кх8 ОЗУ

Урок 2

СИСТЕМЫ СЧИСЛЕНИЯ

В этом разделе рассматриваются десятичная, двоичная, восьмеричная и шестнадцатеричная системы счисления. Человек обычно работает с десятичной системой, а цифровая ЭВМ - с двоичной системой. Поскольку наиболее легкое представление двоичных величин обеспечивают восьмеричная и шестнадцатеричная системы, "Микролаб" использует шестнадцатеричную систему для пред-

ставления двоичнозакодированных программ и данных.

Рассмотрим общее представление чисел в различных системах счисления.

ДЕСЯТИЧНАЯ И ДВОИЧНАЯ СИСТЕМЫ СЧИСЛЕНИЯ

Наиболее привычной для оператора является десятичная система или система с основанием 10. Основание 10 означает, что одним цифровым знаком можно представить десять различных чисел: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Цифровые ЭВМ используют двоичную систему или систему с основанием 2. Это значит, что одной цифрой можно представить два различных числа: 0 и 1. Это естественная система счисления для цифровых систем, поскольку цифровой сигнал имеет только два состояния: низкое и высокое, которые могут быть интерпретированы, как 0 и 1.

В табл.1 сравниваются десятичная и двоичная системы счисления. В десятичной системе позиции цифр имеют следующие веса: $10^0=1$, $10^1=10$, $10^2=100$, $10^3=1000$ и т.д. В двоичной системе $2^0=1$, $2^1=2$, $2^2=4$ и т.д. Чтобы перейти из двоичной системы в десятичную, десятичные числа в строчке просто складываются: $11001=/1x2^0/+/0x2^1/+/0x2^2/+/1x2^3/+/1x2^4/=1+0+0+8+16=25$.

В табл.2. приведены двоичные эквиваленты десятичных цифр от 0 до 31 вместе с шестнадцатеричными и восьмеричными представлениями. Рис.1 показывает механизм преобразования числа из десятичной в двоичную систему. Сначала десятичное число делится на 2, остаток от деления /который в этом примере равен 1/ становится младшим разрядом результата /крайним справа/. Результат деления затем снова делится на 2, остаток будет вторым разрядом результата. Этот процесс продолжается до тех пор, пока в результате частное от деления не станет равным нулю.

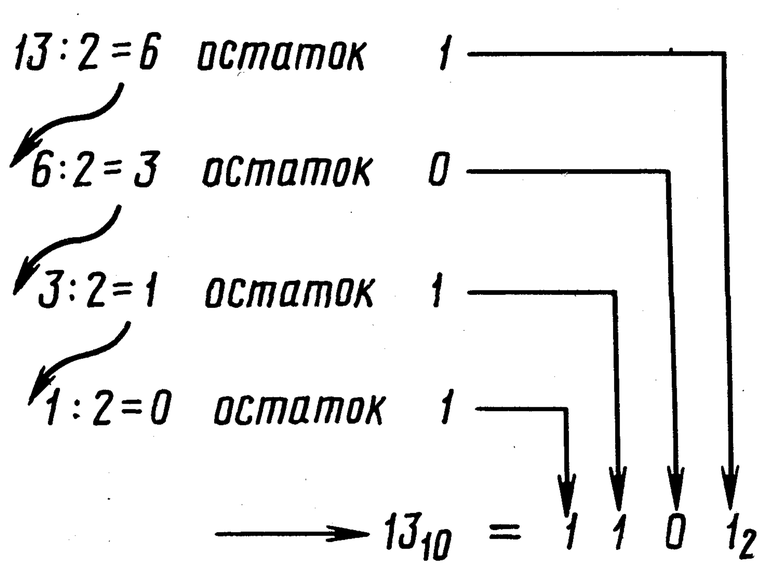


Рис.1. Преобразование числа из десятичной в двоичную систему

Таблица 1

Десятичная и двоичная системы счисления

Вес каждого разряда	Десятичная система					Двоичная система				
	10^4	10^3	10^2	10^1	10^o	2^4	2^3	2^2	2^1	2^o
Пример	1	9	6	0	1	1	0	1	1	0
	$19\,601=1x10^4+9x10^3+6x10^2+0x10^1+1x10^o=10000+9000+600+1$					$10110=1x2^4+0x2^3+1x2^2+1x2^1+0x2^o=10+4+2$				
	$10^o=1$					$2^o=1$				
	$10^1=10$					$2^1=2$				
	$10^2=100$					$2^2=4$				
	$10^3=1000$					$2^3=8$				
	$10^4=10000$					$2^4=16$				

Таблица 2

Двоичное, восьмеричное и шестнадцатеричное представление

Двоичная	Десятичная	Восьмеричная	Шестнадцатеричная
00000	0	0	0
00001	1	1	1
00010	2	2	2
00011	3	3	3
00100	4	4	4
00101	5	5	5
00110	6	6	6
00111	7	7	7
01000	8	10	8
01001	9	11	9
01010	10	12	A
01011	11	13	B
01100	12	14	C
01101	13	15	D
01110	14	16	E
01111	15	17	F
10000	16	20	10
10001	17	21	11
10010	18	22	12
10011	19	23	13
10100	20	24	14
10101	21	25	15
10110	22	26	16
10111	23	27	17
11000	24	30	18
11001	25	31	19
11010	26	32	1A
11011	27	33	1B
11100	28	34	1C
11101	29	35	1D
11110	30	36	1E
11111	31	37	1F

ВОСЬМЕРИЧНАЯ СИСТЕМА

Поскольку двоичные числа содержат больше цифр, чем их десятичный эквивалент, они более трудны в обращении. Например, при записи числа "10111001" легко сделать ошибку и записать "10110001". Чтобы избежать подобных ошибок, необходимо более компактное представление двоичных чисел. Можно было бы использовать десятичные числа, но преобразование из десятичной в двоичную систему очень неудобно. На рис.2 показано представление, называемое восьмеричной системой, т.е. системой с основанием 8.

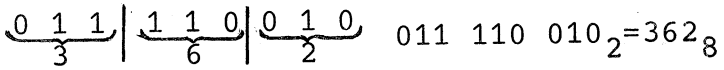


Рис.2. Восьмеричная система счисления

Двоичное число делится на группы по три разряда каждая, начиная справа. Каждая такая группа заменяется ее восьмеричным эквивалентом. Поэтому двоичное число "101001" может быть представлено как восьмеричное число 51. Однако, хотя восьмеричные числа и выглядят, как десятичные, их нельзя путать. Десятичный эквивалент числа 101001 есть $2^0+2^3+2^5=1+8+32=41$. Гораздо легче произвести прямые и обратные преобразования между двоичной и восьмеричной системами, чем между двоичной и десятичной.

ШЕСТНАДЦАТЕРИЧНАЯ СИСТЕМА СЧИСЛЕНИЯ

Другим представлением двоичных чисел является их представление в шестнадцатеричной системе счисления или системе с основанием 16. Каждая группа из четырех разрядов заменяется одним символом /рис.3/.

Так как четыре разряда занимает представление чисел от 0 до 15_{10}, необходим способ для представления десятичных величин от 10 до 15. Для этой цели используют буквы от A до F. В шестнадцатеричной системе /как показано на рис.3/ счет ведется так: 0, 1...8, 9, A, B, C, D, E, F, 10, 11,... Преобразования между двоичной и шестнадцатеричной системами очень легки. Двоичное число делится на группы по четыре разряда, и каждая группа преобразуется в соответствующий символ. Чтобы произвести обратное преобразование, каждый символ заменяется четырьмя разрядами. Привыкнув использовать букву A для обозначения 10, B - для 11 и т.д., можно выполнять преобразования очень легко. Шестнадцатеричная система, наиболее широко употребляемая, будет использована в данном учебном курсе. Она более предпочтительна, чем восьмеричная, поскольку восьмиразрядное двоичное число может быть представлено двумя шестнадцатиразрядными символами и в то же время

тремя восьмиразрядными. Поэтому шестнадцатеричное представление чисел более компактно.

1 1 1 1 | 0 0 1 0 $1111\ 0010_2=F2_{16}$

F 2

Двоичная	Шестнадцатеричная	Дисплей
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	A
1011	B	b
1100	C	C
1101	D	d
1110	E	E
1111	F	F

Рис.3. Шестнадцатеричная система счисления

ТЕРМИНОЛОГИЯ РАЗРЯДНЫХ ПОЗИЦИЙ

При работе с двоичными числами иногда необходимо рассмотреть отдельный бит или группу битов. Крайний бит справа называется младшим разрядом, а крайний левый бит называется старшим разрядом. При работе с группой битов крайний левый бит в слове называется разрядом наивысшего порядка или наиболее значащим. Крайний бит справа называется битом наинизшего порядка или наименее значащим битом.

Урок 3

ОСНОВЫ МАТОБЕСПЕЧЕНИЯ

Как описано в предыдущем разделе, система микро-ЭВМ состоит из аппаратной части и матобеспечения. Программа пишется сначала в таком виде, который удобен для программиста, а затем она должна быть переписана и храниться в кодах, "понятных" МП. МП затем считывает эти коды из памяти и выполняет указанные операции. Важно понять, что может делать микро-ЭВМ и чего она делать не может.

ЧЕГО НЕ "ПОНИМАЕТ" МИКРО-ЭВМ

При составлении программы программист должен дать микро-ЭВМ детальное описание ее действий. Микро-ЭВМ может действовать с большой точностью и скоростью в течение длительного периода времени, но ей необходимо дать четкое задание. Она может реагировать на изменение условий, но только в том случае, если введена программа, сообщая ей: "В данных условиях нужно сделать следующее". Компьютеры действуют логически, но не обладают творческими способностями. Кажущаяся разумность микрокомпьютеров является всего лишь результатом большого числа программ, заложенных в них.

МИКРО-ЭВМ КАК ЛОГИЧЕСКОЕ УСТРОЙСТВО

Микропроцессорные системы часто используются для замены схем, состоящих из стандартных логических устройств. Чтобы проиллюстрировать разницу между "запрограммированными" и традиционными логическими устройствами, рассмотрим использование МП в качестве простой схемы "И". Схема "И", основанная на МП, требует входного порта для входов схемы и выходного порта для выходов схемы /рис.1/. МП, используя команды, хранящиеся в памяти, выполняет функцию "И". Поскольку схема "И" имеет только один выход, требуется только один разряд в выходном порте.

Для схемы "И", основанной на МП, требуется соответствующая прог-

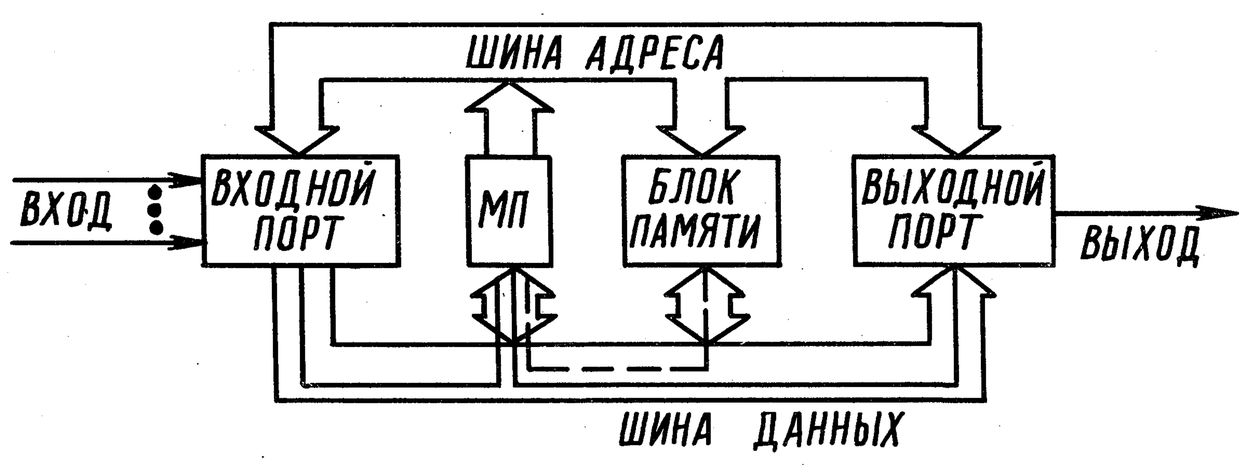


Рис.1. Схема микропроцессора, выполняющего функции схемы И

рамма. Далее приводится список команд, выполняющих функцию "И":

1. Считать данные с входного порта.
2. Перейти на шаг 5, если все входы имеют высокий уровень, в противном случае - продолжать.
3. Установить низкий уровень на выходе.
4. Перейти на шаг 1.
5. Установить высокий уровень на выходе.
6. Перейти на шаг 1.

Сначала считываются данные с порта входа. Затем данные проверяются на высокий уровень. Если на всех входах уровень высокий, на выходе устанавливается тоже высокий уровень, в противном случае устанавливается низкий уровень. Когда программа завершена, происходит переход на шаг 1 и повторение. При этом изменения на входах постоянно отражаются на выходе.

Графически программа представлена структурной схемой, состоящей из блоков различных типов, соединенных линиями /рис.2/: прямоугольный блок обозначает любые действия

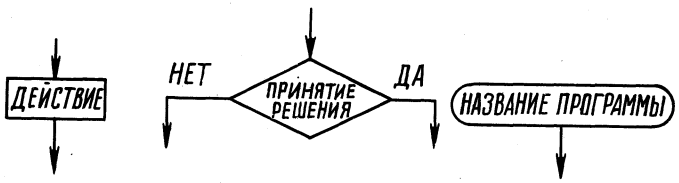


Рис.2. Основные типы блоков

программы, ромбический - принятие решения. Овальный блок ставится в начале схемы, в нем записывается название программы,а также в конце схемы.

На рис.3. показана структурная схема программы для схемы "И". Стрелки показывают направление выполнения программы. Хотя структурная схема содержит ту же информацию, что и листинг /распечатка/ программы, она более наглядна.

Если составление программы затруднительно, структурная схема поможет продумать, как она должна быть написана. Используя структурную схему, можно проверить логику рассуждений, вернуться к ранее написанной программе и быстро восстановить ее функции.

Проведем эксперимент, в котором "Микролаб" выступает в роли

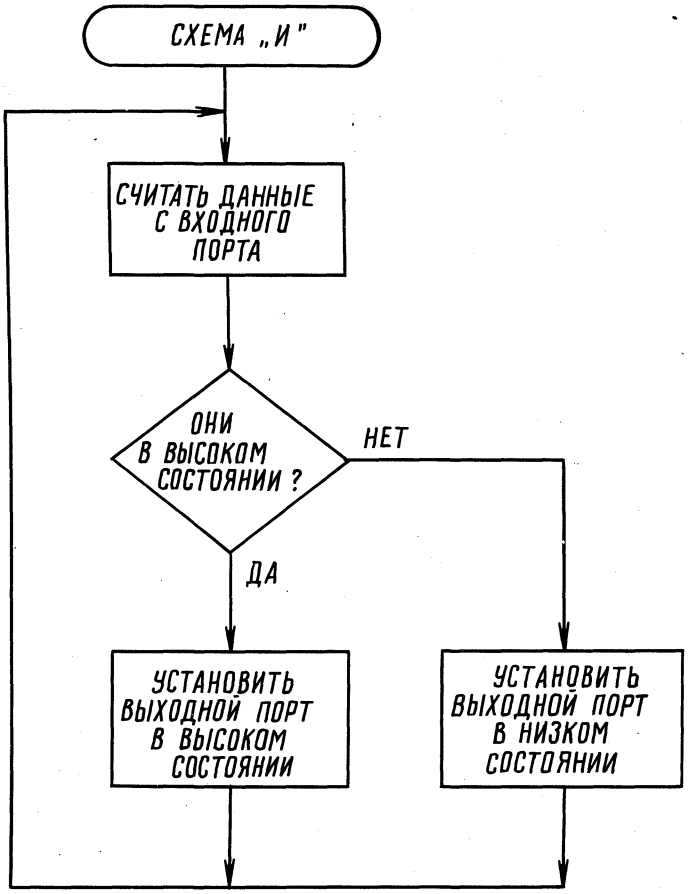


Рис.3. Структурная схема программы И

схемы "И", а ПЗУ микро-ЭВМ содержит программу для выполнения функции "И". Эксперимент позволит пронаблюдать за выполнением данной программы. Для этого необходимо:

1. Нажать кнопку СБРОС,
2. Нажать кнопки 0, 3, Е, 0 и УСТ.АД., тем самым установить начальный адрес программы,
3. Нажать кнопку ПУСК - программа "И" начнет выполняться.
4. Найти три тумблера, помещенные внизу панели. Эти тумблеры связаны с портом ввода и являются входами схемы "И" /расположенные рядом восемь светодиодов связаны с портом вывода. Самый правый используется как выход схемы "И"/,
5. Установить все три входных переключателя в верхнее положение, тем самым подать высокий уровень на разряды входного порта. Крайний справа светодиод будет светиться, так как на всех входах высокий уровень. Этот индикатор связан с нулевым битом порта вывода, который используется как выход схемы "И",
6. Изменить положение любого из входных переключателей. Индикатор

перестанет светиться, поскольку не на всех входах будет высокий уровень.

7. Нажать кнопку СБРОС. Установить входные переключатели в верхнее положение. На выходе /светодиоде/ не будет никакого отклика. Это значит, что выполнение программы прекратилось.

Микропроцессорная система может выполнять функцию обычной схемы "И" с одним недостатком - она менее быстродействующая. Программа схемы "И" состоит из четырех команд, а "Микролаб" требует около 3 мкс для выполнения одной команды. Поэтому схема "И" имеет время задержки /время между изменениями на входе и на выходе/, равное примерно 3 мкс х 4 = 12 мкс. Обычная ТТЛ схема имеет время задержки приблизительно 10 нс - - более чем в 1000 раз меньшее. Этот пример иллюстрирует быстродействие МП в сравнении с обычной логикой. Разница тем больше, чем сложнее система. Однако часто это не столь существенно, так как скорость обычной логики в действительности гораздо больше, чем требуется. Для чего использовать такую сложную систему, если достаточно обыкновенной микросхемы "И"? Действительно, если это единственная функция, которую система должна выполнять, нужно использовать схему "И". Однако МП обеспечивает большую гибкость. На его основе можно построить различные схемы с помощью изменения программы, можно добавить значительно больше входов,и функции схемы намного усложнятся. Рассмотрим, например, 8-входовое логическое устройство, которое действует как электронный замок. Выход активизируется, если входы расположены в специальном порядке. Это можно осуществить, используя традиционную логику, а также микропроцессорную систему, как для простой схемы "И". Конечно, требуется новая программа, более сложная, чем программа схемы "И", но аппаратная часть, кроме устройств ввода/вывода, не изменится. Кроме того, "код" электронного замка может быть изменен за счет изменения программы.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Язык, "понятный" МП, на котором пишутся программы, называется машинным языком /машинным кодом/. Так как МП имеет дело непосредственно с цифровыми сигналами, то команды машинного языка закодированы в двоичном коде, например 00111100. Человеку трудно пользоваться машинным языком, поскольку запись 00111100 - мало наглядна. Можно облегчить работу, употребляя шестнадцатеричную систему, заменяя 00111100 на "3С". Однако этого еще недостаточно для распознавания команды. Следующий шаг - замещение командного кода коротким именем, называемым мнемоникой. Например, код "3С" для МП КР580ИК80 означает увеличение содержимого регистра Н и обозначается как INR A. Мнемоника гораздо легче для запоминания, чем машинные коды. Заменяя каждый код мнемоникой, можно написать программу. Мнемоника может быть легко преобразована в машинные коды после написания программы. Вот почему не обязательно запоминать машинные коды, а мнемоническое значение каждой команды запомнить легко. Программы, написанные с использованием мнемоники, называют программами на языке Ассемблер. Машинный язык обычно определяется назначением МП и не может быть изменен. Ассемблер создается при производстве МП для удобства программиста, но не предусматривается назначением устройства. Например, можно записать INC A вместо INR A, и оба выражения будут переведены в машинный код "3С". Хотя Ассемблер имеет большие преимущества перед машинным языком, он труден при написании сложных программ. Чтобы облегчить программирование, были созданы языки высокого уровня. Они похожи на английский язык и совершенно не зависят от типа МП. Типичной командой может быть LET COUNT=10 или PRINT COUNT. Эти команды предполагают гораздо более сложные действия, чем то, что может "понять" МП. Вот почему микро-ЭВМ, использующие языки высокого уровня, также содержат длинную сложную программу /постоянно хранящуюся в их памяти/, которая переводит программу с языка высокого уровня на машинный язык. Простые команды, написанные на высокоуровневом языке, могут переводиться в десятки ко-

манд на машинном языке. Такой транслятор программы называется компилятором.

ПРИМЕРЫ ПРОГРАММИРОВАНИЯ

На рис.4 приведена структурная схема программы, считающей до 10. В этой программе нет начала и конца, содержимое ячейки просто нарастает от 0 до 10 и затем все повторяется /виды программ и программирование будут более детально обсуждаться в следующих разделах/.

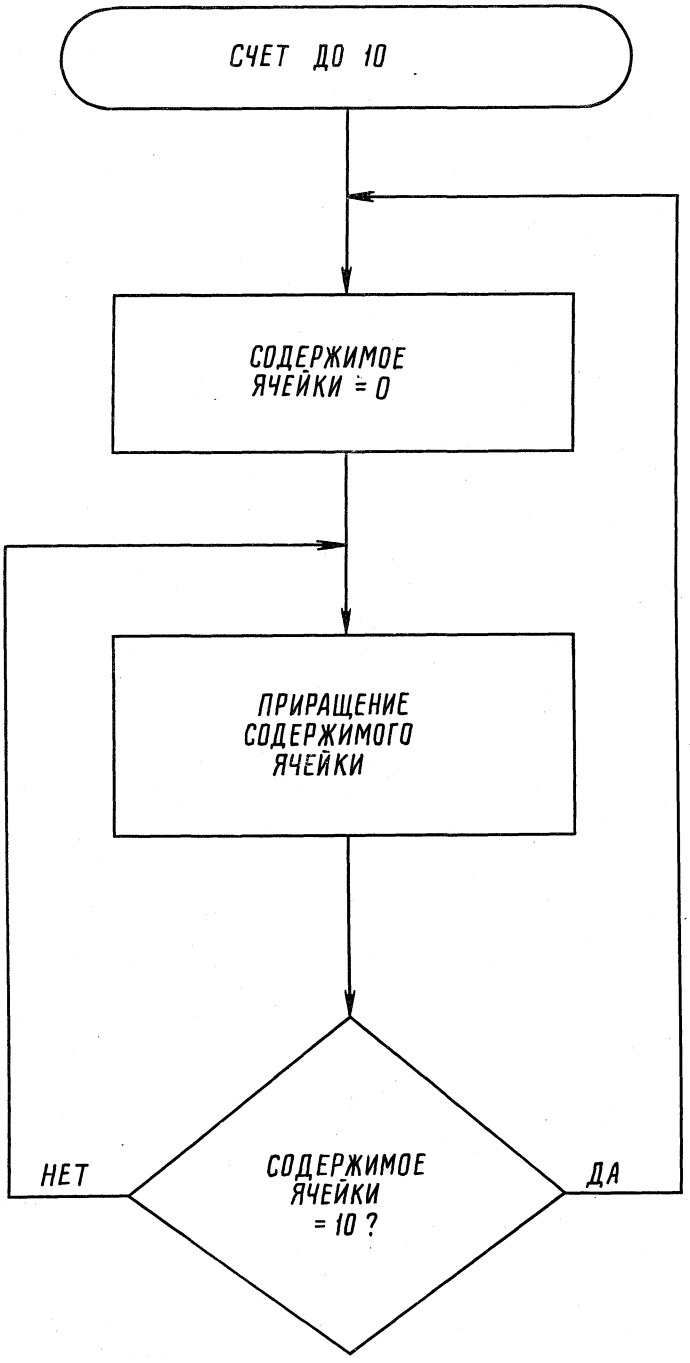


Рис.4. Структурная схема программы, считающей до 10

Переход от структурной схемы к языку высокого уровня очень прост. Следующий пример показывает вариант высокоуровневого языка, называемого BASIC, обладающего преимуществом, заключающимся в простоте и схожести с английским языком.

В табл.1 приводится листинг программы.

Таблица 1

Листинг программы счета до 10 на языке BASIC

1. LET	COUNT=0	Установка ячейки в 0
2. LET	COUNT=COUNT+1	Приращение ячейки
3. IF	COUNT=10 THEN 1	Перейти на 1, если ячейка=10
4. GO	TO2	Перейти на 2

Первые строчки программы точно соответствуют двум первым блокам схемы. В первой строчке в ячейку памяти, обозначенную COUNT, засылается 0. Вторая строчка LET COUNT= COUNT+1 - это просто способ обозначения счета. Третья и четвертая строчки выполняют функции проверочного блока. Третья строка проверяет, равняется ли COUNT 10. Если COUNT=10, то выполняется команда, записанная в первой строке. Если же COUNT≠10, то эта команда не выполняется, а выполняется строка 4. Эта команда говорит: "Перейди на строку 2". Таким образом, две эти команды выполняют действия, требуемые блоком проверки в структурной схеме. Если проследить программу шаг за шагом, можно увидеть, как происходит накопление результата счета.

Каждый МП имеет собственный машинный язык, т.е. собственный Ассемблер. В данном примере используется Ассемблер МП КР580ИК80, на котором разработана "Микролаб".

В табл.2 приведен листинг программы счета до 10 на языке Ассемблер.

Таблица 2

Листинг программы счета до 10 на языке Ассемблер

```
START: MVI A,0    ;Установить А регистр в 0
LOOP:  INR A      ;Прирастить А регистр
```

```
CPI 10      ;Сравнить А регистр
             с 10
JZ START    ;Перейти на начало,
             если А=10
JMP LOOP    ;Перейти на LOOP
```

Эта программа менее понятна, чем та же программа на языке BASIC, хотя выполняет ту же функцию, так как BASIC ближе к английскому языку, чем к машинному языку МП. Три столбца - это метки, команды и комментарии. Метка выполняет ту же функцию, что и номер строки. Вместо нумерации каждой строки нужно выбрать метку /имя/ для каждой строки, на которую необходимо будет сослаться. Символ /;/ используется для распознавания метки. Метка в строке необходима, только если в программе присутствует команда, ссылающаяся на эту строку. Метка позволит легче распознать строку, на которую вы хотите перейти при выполнении программы. Комментарии служат для облегчения восприятия программы. Символ /;/ идентифицирует начало комментария. Программа на высокоуровневом языке не требует большого числа комментариев, поскольку команды сами по себе понятны.

Для программы на языке Ассемблер, однако, комментарии необходимы. Они полезны для всех, кому нужно понять программу, или для программиста, которому надо вернуться к этой программе по истечении некоторого времени. Первая команда MVI A, 0 /непосредственная засылка в аккумулятор 0/. Аккумулятор /называемый А-регистром/ - это место хранения данных в МП. Эта команда эквивалентна команде LET COUNT=0, за исключением того, что вместо имени переменной COUNT используется обозначение А для внутреннего регистра МП. Позднее будет сказано, почему этот регистр называется аккумулятором. Сейчас для простоты примем, что это место в МП, куда загружаются нулевые данные. Следующая команда INR A означает увеличение содержимого аккумулятора. Аккумулятор содержит результат счета, так что эта команда эквивалентна команде LET COUNT=COUNT+1. Все три следующие команды выполняют функцию проверки. Команда CPI 10 /непосредственное сравнение/ означает сравнение величины в аккумуляторе с числом 10. Она не вызывает перехода в зависимости от результата сравнения, а вместо этого устанавливает специальный триггер, называемый флагом, если величина в регистре А равна 10. Следующая команда JZ START проверяет этот флаг. Если числа равны, эта команда определяет, что флаг установлен, и вызывает переход на строку с меткой START. Обе эти команды /CPI 10 JZ START/ выполняют функцию оператора IF COUNT≠0 THEN 1 на языке BASIC.

Последняя команда JMP LOOP просто обеспечивает переход на строку с меткой LOOP.

Это эквивалентно оператору GO TO2 на языке BASIC.

Рассматриваемая программа в машинных кодах приведена в табл.3.

Таблица 3

Программа счета до 10 на трех языках

Basic	Ассемблер	Машинный язык	
1. LET COUNT=0	START: MVI A,0	адрес	содержимое
		8000	3E код
		8001	00 данные
2. LET COUNT=COUNT+1	LOOP: INR A	8002	3C код
3. IF COUNT=10 THEN1	CPI 10	8003	FE код
		8004	OA данные
	JZ START	8005	CA код
		8006	00 адрес
		8007	80 адрес
4. GO TO 2	JMP LOOP	8008	C3 код
		8009	02 адрес
		800A	80 адрес

Функции этой программы не изменились, а особенности работы с машинным языком теперь очевидны. Однако причина использования этого языка существенна: МП "понимает" только его. Отметим, что содержимое ячеек памяти приведено в таблице в шестнадцатеричном коде, а МП "понимает" только двоичный код. Значит шестнадцатеричный машинный код должен быть переведен в двоичный.

Каждая ячейка памяти в "Микролабе" содержит 8 битов информации /что может быть представлено двумя шестнадцатеричными символами/. Каждая команда начинается с кода операции, необходимого для выполнения команды. Все коды операций /КОП/ для КР580ИК80 имеют 8 битов в длину и занимают одну ячейку памяти. За КОП могут последовать 0; 1 или 2 байта данных в зависимости от команды. Первый байт /3E/ по адресу 8000 - это КОП для команды MVI A. Однако это только часть всей команды. Он информирует о том, что необходимо поместить некоторую информацию в аккумулятор. Далее требуется другая ячейка памяти для загрузки самой этой информации. Следовательно, следующая ячейка памяти /адрес 8001/ содержит 00 - данные, необходимые для засылки в аккумулятор. Третья ячейка содержит КОП второй команды INR A. Этот КОП /3C/ "приказывает" МП увеличить содержимое аккумулятора. Так как в этой команде не имеется добавочной информации, она занимает только одну ячейку памяти. Код FE - это КОП для команды сравнения CPI. Как и в команде MVI A, здесь за КОП следуют добавочные данные для ее выполнения, которым отводится еще одна ячейка памяти. Вследствие шестнадцатеричного представления информации число 10 выражается как 0A. Команда сравнивает содержимое аккумулятора с числом 10 и устанавливает флаг /как это было описано выше/, если они равны. Команда JZ имеет КОП CA, который имеет адрес 8005. Этот КОП информирует МП о необходимости перехода, если флаг установлен. В следующих двух ячейках памяти хранится адрес, на который нужно перейти. Поскольку длина адреса для КР580ИК80 16 битов, требуется две ячейки /по 8 битов/ для хранения адреса. Части адреса хранятся в обратном порядке: младшие разряды записываются в первую очередь, старшие - следом за ними. Таким образом, адрес 8000 хранится в виде 00 80. Команда Ассемблера JZ START означает, что МП должен перейти к команде с меткой START. Далее программа должна считывать КОП из ячейки, соответствующей метке START /8000 в нашем случае/. Последняя команда JMP LOOP закодирована таким же образом. Разница лишь в том, что этот переход не зависит от каких-либо условий. Код такого типа перехода - C3. Адрес перехода /8002/ хранится в ячейках памяти так же, как и для команды JZ.

Отметим, что программа на машинном языке состоит из последовательностей байтов, каждый из которых может иметь одно из трех значений: КОП, данные, адрес перехода. Электрические схемы МП определяют, что должно следовать за КОП - либо данные, либо адреса, и "помнят" об этом.

Из вышесказанного следует, что языки высокого уровня гораздо легче для программиста и не зависят от частного вида МП. Однако переведенные в машинный код программы занимают много места в памяти; кроме того, нужно иметь длинные программы - трансляторы. Высокоуровневые языки также менее эффективны в смысле скорости выполнения операций и распределения памяти. Эквивалентная программа, написанная на языке Ассемблер, выполняется значительно быстрее и занимает меньше памяти. Ассемблер широко используется для программирования на микро-ЭВМ. Он более труден для написания программ, чем язык высокого уровня, однако он легче для перевода в машинные коды. Там, где необходимо быстрее и экономнее выполнить программы, Ассемблер незаменим. Программирование на Ассемблере поможет овладеть навыками работы с микропроцессорными системами.

Машинный язык - единственный, непосредственно воспринимаемый МП, но требуется много времени для овладения им. Обычно программы пишутся на языке Ассемблер, а затем транслируются в машинные коды. Трансляция выполняется специальной программой, называемой Ассемблер.

Таким образом, непосредственно в "Микролаб" вводятся программы

только на машинном языке. Для машинных кодов используется шестнадцатеричное представление как более легкое по сравнению с двоичным. "Микролаб" имеет матобеспечение для сравнительно простого перевода из шестнадцатеричного в двоичный код.

Проведем эксперимэнт по проверке содержимого части памяти "Микролаб".Для этого необходимо:

1. Нажать кнопку СБРОС;

2. Нажать кнопку 5, 5, УСТ.АД. Это означает, что надо проверить адрес 0055. Данные, которые хранятся по этому адресу /31/, можно увидеть на двух крайних правых индикаторах;

3. Для проверки следующей ячейки нажать кнопку АД +. На тех же двух правых индикаторах можно увидеть содержимое следующей ячейки;

4. Нажимая кнопку АД. +, просмотреть часть программы и сравнить ее с распечаткой программы монитора, приведенной в техническом описании "Микролаб";

5. Проследить в распечатке монитора соответствие машинных кодов и мнемоники /коды команды приведены в приложении/.

Урок 4

ИСПОЛЬЗОВАНИЕ "МИКРОЛАБ КР580ИК80"

Микролаборатория - микро-ЭВМ, разработанная специально для целей обучения. Она использует МП КР580ИК80, ПЗУ емкостью IK и ОЗУ емкостью IK. В микро-ЭВМ имеется клавиатура, с которой можно вводить программы, данные для хранения, подавать команды для управления работой микрокомпьютера и дисплея, позволяющего наблюдать содержимое памяти и регистров, восемь светодиодных индикаторов и три тумблера, связанных с МП посредством порта ввода/вывода. Генерация звуков также производится процессором. ПЗУ содержит программы для считывания с клавиатуры, выполнения команд и вывода данных на дисплей, т.е. вся работа системы управляется программой монитора, записанной в ПЗУ.

Хранение данных в памяти, проверка содержимого памяти и исполнение программ - основные операции, выполняемые микролабораторией. Программы могут выполняться непрерывно или в шаговом режиме, что позволяет проследить их работу подробно.

В табл.1 приведена карта памяти микролаборатории. На ней показано распределение адресов памяти для каждого устройства. ПЗУ резервирует адреса с 0000 по 03FF. Монитор использует адреса с 0000 по 02FF, а 0300-03FF /265 байт/ могут быть использованы для расширения возможностей монитора. Область с 0400 до 05FF также может быть использована для записи дополнительных программ. Следует только помнить, что программа в эти области может быть записана только программированием ПЗУ. Программа, которая занимает область адресов с 0400 до 05FF, записывается /путем программирования/ в ПЗУ, находящееся в адаптере.

ОЗУ резервирует адреса с 8000 по 83FF. Заметим, однако, что не все ОЗУ доступны для наших программ. Адреса с 83C7 до 83FF используются программой монитора и не могут быть использованы нашими программами. Область пользователя: с адреса 8000 по 83C6.

Таблица 1

Карта памяти "Микролаб КР580ИК80"

Адреса	Емкость памяти	ПЗУ/ОЗУ	Использование
FFFF — 8400	31K		Неиспользуемая область
83FF — 83C7	57	ОЗУ	Рабочая область монитора
83C6 — 8000	967	ОЗУ	Область пользователя
7FFF — 0600	30,5K		Неиспользуемая область
05FF — 0400	512	ПЗУ	Область пользователя
03FF — 0300	256	ПЗУ	Дополнительная область монитора
02FF — 0000	768	ПЗУ	Область монитора

ПРОВЕРКА ПАМЯТИ, ЗАПИСЬ И ХРАНЕНИЕ ДАННЫХ

Проведем эксперимент по проверке содержимого памяти.

1. Нажмите кнопку СБРОС.

2. Нажмите кнопки 8, 0, 0, 0 и УСТ.АД. Теперь четыре левых цифры дисплея показывают адрес, который только что ввели, а две крайние правые цифры показывают данные, хранящиеся по этому адресу. Адреса и данные, высвечиваемые на дисплее, выражены в шестнадцатеричном коде.

3. Нажмите кнопку АД.+. На левых индикаторах будет наблюдаться приращение адреса, а на двух крайних правых индикаторах высветится содержимое следующей ячейки памяти. Многократно нажимая кнопку АД.+, можно просмотреть содержимое памяти.

4. Нажмите кнопку АД.-. На левых индикаторах произойдет уменьшение адреса на 1, а на двух крайних правых индикаторах высветится содержимое ячейки памяти, соответствующей этому адресу. Таким образом, нажимая кнопку АД.-, можно просмотреть содержимое памяти в обратном порядке.

Проведем эксперимент по записи данных в память "Микролаб".

1. Нажмите кнопки 8, 0, 0, 0 и УСТ.АД. Это определит адрес 8000, который является начальным адресом ОЗУ.

2. Нажмите кнопки 0, 0, ЗП, т.е. запишите 00 в ячейку памяти с адресом 8000. Произошло приращение адреса, и на левых индикаторах виден адрес 8001. На двух крайних правых индикаторах теперь видно содержимое ячейки с адресом 8001, а то, что было записано в ячейку с адресом 8000, сместилось на два индикатора левее.

3. Нажмите кнопки С, З и ЗП, т.е. запишите данные СЗ по адресу 8001. Происходит приращение адреса.

4. Запишите в следующую ячейку с адресом 8002 данные 00, а в ячейку с адресом 8003 - 80.

5. Нажмите кнопку АД.-. Это уменьшит на единицу адрес и на двух крайних правых индикаторах появятся данные, которые только что ввели.

6. Повторите шаг 5, пока не проверите, что весь введенный вами листинг содержится в памяти.

Можно также проверить данные, нажав кнопки 8, 0, 0, 0 и УСТ.АД. и затем используя кнопку АД.+ для проверки последовательных ячеек.

Проведем эксперимент по исправлению ошибок.

1. Нажмите кнопки 0, 0, 0,1 и УСТ.АД. - выбрали первый адрес ПЗУ.

2. Нажмите кнопки 0, 0 и ЗП. Это означает, что вы пытаетесь записать нули в выбранную ячейку, а эта ячейка находится в ПЗУ и данные туда не запишутся.

3. Чтобы убедиться в этом, проверьте содержимое ячейки с адресом 0001. Для этого нажмите кнопку АД.-. Вы увидите, что в ячейке с адресом 0001 записано 92, а не 00. Таким образом, ваше ошибочное действие не было воспринято.

Теперь рассмотрим, как быть с такими ошибками, как ввод неправильных адресов или данных. Проведем следующий эксперимент.

1. Нажмите кнопки 0, 9, 0. Предположим, что в этот момент вы вспомнили, что адрес, который вы хотите записать, 8020. И, хотя вы уже начали вводить адрес, ошибку можно исправить, нажав кнопки 8, 0, 2, 0 и УСТ.АД.

2. Теперь, когда вы ввели правильный адрес, можно ввести нужные данные. Нажмите кнопки 7, 9. Эти данные появятся на правой стороне дисплея.

3. Предположим, что это не те данные, которые вы хотели ввести: вам нужно было ввести 69. Тогда нажмите кнопки 6, 9 и введите правильные данные, а неправильные сместятся левее и будут утеряны. Можно вводить данные непрерывно, но "Микролаб" сохраняет только две последние цифры. После этого, если нажать кнопку ЗП, данные запишутся в память.

4. Нажмите кнопку ЗП. Теперь записали данные 69 по адресу 8020.

5. Предположим, что данные надо изменить на 68. Нажмите кнопку АД.-. На дисплее появится адрес 8020 и данные 69. Наберите нужные данные /68/ и нажмите кнопку ЗП. Таким образом данные изменены.

ПРОСТАЯ ПРОГРАММА

Данные, записанные в ОЗУ, приведенные в табл.2, являются реальной программой.

Таблица 2

Данные, записанные в ОЗУ

Адрес	Содержимое
8000	00
8001	C3
8002	00
8003	80

Код 00, хранимый по адресу 8000, имеет мнемоническое обозначение NOP, которое определяет отсутствие операции. Эта команда, как указывает ее имя, ничего не делает /не выполняет никаких действий/. Однако она может быть использована в качестве короткой временной задержки. Но наиболее важно использование ее в качестве пространственного заполнителя. Если позднее надо вернуться назад и ввести новые команды в программу, которая уже написана, можно заменить команды NOP нужными командами. Если вы не поместили команды NOP в программу, вам придется сдвигать большой кусок программы для того, чтобы поместить одну дополнительную команду.

Другая команда программы - переход /JMP - мнемоническое обозначение/. Код операции - C3. Эта команда означает переход к адресу, определяемому в следующих двух байтах памяти. В первом байте памяти содержится младший адресный байт, а во втором - старший адресный байт. Таким образом, адрес перехода 8000 хранится следующим образом: 00 - в ячейке 8002 и 80 - в ячейке 8003.

Эта программа, конечно, ничего не делает, но она обеспечивает бесконечный цикл. Структурная схема программы показана на рис.1.

Проведем эксперимент по выполнению программы пользователя в автоматическом и шаговом режимах.

1. Переключите тумблер режима в положение ШАГ. Программа будет выполняться в шаговом режиме.

2. Введите программу, указанную в табл.2.

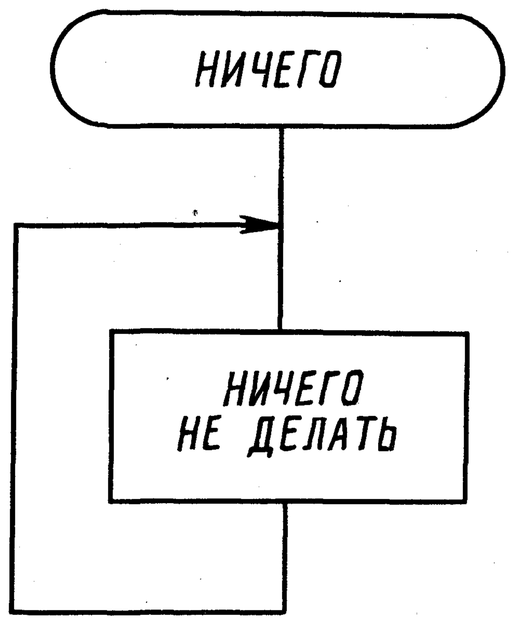


Рис.1. Структурная схема программы "ничего не делать"

3. Установите начальный адрес программы 8000.

4. Нажмите кнопку ПУСК. Команда, показанная на дисплее, выполнится, и дисплей покажет следующую команду. Заметим, что, хотя может показаться, что кнопка ПУСК /в шаговом режиме/ работает так же, как кнопка АД.+, их функции совершенно различны. Когда используете кнопку АД.+, вы просто проверяете содержимое памяти. Нажимая кнопку ПУСК /в шаговом режиме/, вы интерпретируете содержимое ячейки памяти, высвеченное на дисплее, как команду, которая выполняется МП.

Перед тем, как "Микролаб" выполнит команду, дисплей должен показывать ячейку, которая содержит код операции. Ячейка, с которой начинается выполнение программы, не должна содержать данные или адрес. Например, программа /табл.1/ может начинаться с адресов 8000 или 8001, но не с адресов 8002 или 8003.

5. Теперь на дисплее показана команда перехода C3. Нажмите кнопку ПУСК. Так как это команда перехода, программа перейдет к адресу 8000. Заметим, что ячейки памяти, содержащие адрес перехода, в этом случае не высвечиваются, потому что, когда вы нажимаете кнопку ПУСК /в шаговом режиме/, "Микролаб" выполняет целиком всю команду, включающую считывание двух байтов в адреса перехода.

6. Нажмите кнопку ПУСК еще два раза, чтобы цикл повторился.

7. Переключите тумблер режима в положение АВТ. Теперь программа будет выполняться в автоматическом режиме.

8. Установите начальный адрес 8000 и нажмите кнопку ПУСК. Программа выполняется /непрерывно/ на максимальной скорости /приблизительно 3 мкс на команду/. На дисплее ничего не меняется, так как при выполнении данной программы /табл.2/ не предусмотрен вывод данных на индикацию. /Как и какими командами происходит вывод данных на индикацию, будет рассмотрено ниже/. "Микролаб" не реагирует также при выполнении данной программы на нажатие всех кнопок, кроме кнопки СБРОС. Это происходит вследствие того, что в данной программе не предусмотрен опрос клавиатуры. А из программы монитора, в которой такой опрос предусмотрен, мы вышли в момент нажатия кнопки ПУСК.

9. Нажмите кнопку СБРОС. Заданная программа прекратит исполнять-ся, и начнет выполняться программа монитора. На индикаторах высветятся все нули, и "Микролаб" будет готов к вводу данных.

В отличие от всех остальных кнопок, кнопка СБРОС не опрашивается программным способом, а сигнал с нее заведен на вход СБРОС МП.

ПОРТ ВВОДА/ВЫВОДА

"Микролаб" имеет порт ввода/вывода, который реализован на программируемом периферийном интерфейсе /микросхема КР580ИК55/. Порты этого интерфейса могут быть либо портами вывода, либо портами ввода, в зависимости от того, как интерфейс запрограммирован. К трем разрядам порта с этого интерфейса подключены тумблеры. Этот порт будет программироваться как порт ввода. А к 8 разрядам порта В подключены светодиодные индикаторы. Этот порт будет программироваться как порт вывода, т.е. МП сможет считывать данные с тумблеров и выводить данные на светодиодные переключатели.

На рис.2. показана структурная схема программы, которая демонстрирует использование этих портов. Эта программа считывает данные из порта ввода и записывает их в порт вывода.

Листинг программы приведен в табл.3.

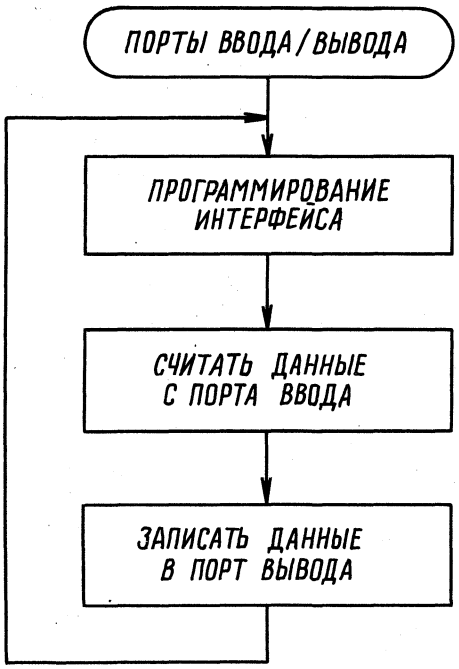


Рис.2. Структурная схема программы для переписи данных из порта ввода в порт вывода

Таблица 3

Листинг программы для переписи данных из порта ввода в порт вывода

Адрес	Содержимое	Метка	Команда	Комментарии
8000	3E		MVI A,81	Запись в аккумулятор данных
8001	81			для программирования интерфейса
8002	D3		OUT FB	Программирование интерфейса
8003	FB			
8004	DB	START:	IN FA	Чтение данных из порта ввода
8005	FA			
8006	D3		OUT F9	Запись данных в порт вывода
8007	F9			
8008	C3		JMP START	Цикл
8009	04			
800A	80			

Чтобы запрограммировать интерфейс, как было предусмотрено /С - порт ввода, В - порт вывода /, необходимо подать на него код 81 по адресу FB. Первая комманда MVI A,81 означает загрузку аккумулятора кодом 81. Следующая команда OUT FB записывает содержимое аккумулятора в порт по адресу FB, т.е. порт /интерфейс/ программируется. Далее идет команда IN FA, обеспечивающая запись данных из порта с адресом FA, присвоенным порту С, в аккумулятор. Команда OUT F9 означает вывод данных из аккумулятора в порт вывода по адресу F9. Поскольку этот адрес в нашем случае присвоен порту В, происходит вывод данных из аккумулятора в порт В /на светодиодные индикаторы/. Таким образом, программа пересылает данные из порта ввода в порт вывода. Последняя команда - переход /JMP/. Она завершает цикл, который выполняется непрерывно. Следовательно, данные выходного порта будут соответствовать данным порта ввода во время выполнения программы. МП КР580ИК80 может адресовать до 256 портов ввода и столько же портов вывода. Поэтому для адресации портов достаточно два шестнадцатеричных разряда, а не четыре, как это требуется для памяти.

Проведем эксперимент по выполнению программы для передачи данных из порта ввода в порт вывода.

1. Введите программу, указанную в табл.3.

2. Начните выполнение программы с адреса 8000.

3. Поставьте тумблер порта ввода в любое положение. Верхнее положение соответствует 1, нижнее - 0.

4. Посмотрите на светодиоды 2-4, считая справа. Они показывают те же данные, которые установлены на тумблерах порта ввода.

5. Измените положение тумблеров порта ввода. Данные порта вывода должны измениться соответственно.

6. Нажмите кнопку СБРОС. Программа остановится, управление перейдет к монитору.

7. Измените положение тумблеров порта ввода. Данные порта вывода не изменятся, так как введенная программа больше не выполняется.

Следует отметить, что, при желании, перед записью данных в порт вывода, можно произвести над ними любые действия, предусмотренные системой команд МП. Например, можно считать данные из порта ввода, проинвертировать их и ввести в порт вывода. Для этого в программу /табл.3/ необходимо добавить команду, обеспечивающую инвертирование данных /CMA/.

Урок 5

ОСНОВНЫЕ ПРИНЦИПЫ ПРОГРАММИРОВАНИЯ

РЕГИСТРЫ МИКРОПРОЦЕССОРА

МП КР580ИК80 содержит ряд внутренних регистров, часть из которых может использоваться для хранения и обработки данных. Эти регистры отличаются от регулярных ячеек памяти тем, что они находятся внутри МП и выбираются конкретной командой /например, команда MVI A выбирает аккумулятор/. Управление регистрами осуществляется непосредственно /без использования шин/ управляющей логикой внутри процессора. Они наиболее подходят для временного хранения и хранения промежуточных результатов.

Некоторые регистры в МП не используются для хранения данных общего назначения, а выполняют специальные функции. Наиболее важным из них является программный счетчик /рис.1/. Это шестнадцатибитный регистр, хранящий адрес команды, которая будет выполняться. Когда команды считываются из памяти, содержимое счетчика команд поступает на шину адреса. Выбираемая

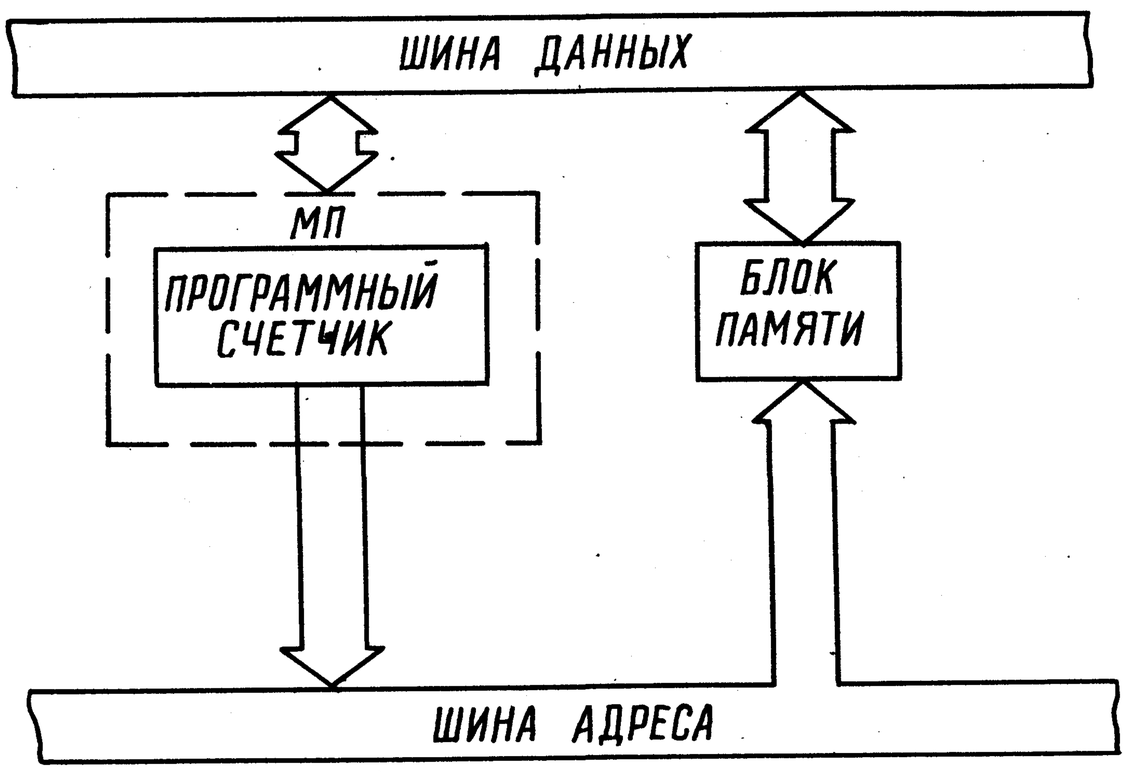


Рис.1. Работа программного счетчика

команда появляется на шине данных. Когда команда будет считана МП, содержимое счетчика увеличится на единицу и снова поступит на шину адреса, затем считывается следующая команда.

ПРОГРАММА МОНИТОРА "МИКРОЛАБ"

ПЗУ микролаборатории содержит программу монитора, которая считывает с клавиатуры, выполняет выбранную операцию и управляет дисплеем. Микролаборатория все время выполняет программу монитора, за исключением случая, когда она выполняет программу пользователя.

Когда нажимается кнопка ПУСК, программа монитора заставляет процессор перейти к адресу, указанному на дисплее "Микролаб". Когда нажимается кнопка СБРОС, микролаборатория возвращается к программе монитора. Программа монитора позволяет проверять содержимое регистров в шаговом режиме после выполнения каждой команды /т.е. после каждого шага/. На третьем и четвертом индикаторах, считая справа, после выполнения каждой команды /шаговый режим/ высвечивается содержимое аккумулятора. Кроме того, после каждого шага команды программа монитора записывает содержимое регистров в специальные ячейки ОЗУ. Следовательно, можно проверить содержимое регистров на каждом шаге, просмотрев соответствующие ячейки ОЗУ /табл.1/.

Таблица 1

Адреса регистров МП

Адрес	Регистр
83EB	Аккумулятор
83EA	Регистр признаков
83E9	B регистр
83E8	C регистр
83E7	D регистр
83E6	E регистр
83E5	H регистр
83E4	L регистр
83E3	Указатель стека /старший байт/
83E2	Указатель стека /младший байт/
83E1	Программный счетчик /старший байт/
83E0	Программный счетчик /младший байт/

ПРОГРАММА СЧЕТА

На рис.2. показана структурная схема программы, которая "заставляет" ячейку памяти считать в двоичном коде от 0 до 255 и затем повторять этот счет.

Сначала один регистр /в данном случае аккумулятор/ устанавливается в 0. Затем содержимое аккумулятора переписывается в ячейку памяти с адресом 8020 и увеличивается на 1. Далее запись в ячейку памяти повторяется снова. В табл.2. приводится листинг программы.

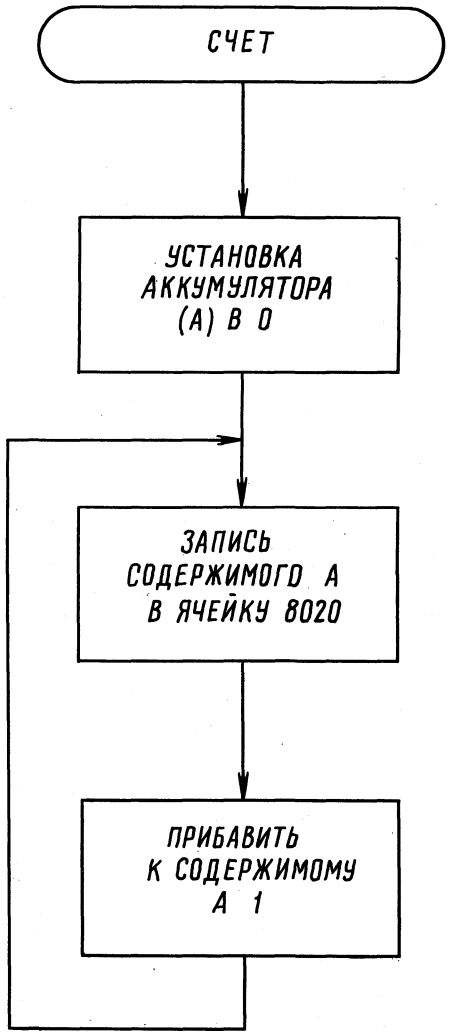


Рис.2. Структурная схема программы счета

Программа начинается с адреса 8004 вместо 8000, так что потом можно добавить несколько команд в начале программы /в "Микролаб" 8000 - первая ячейка ОЗУ, следовательно, программа пользователя не может начинаться раньше этого адреса/. Первая команда - MVI A, 0.

Таблица 2

Листинг программы счета

Адрес	Содержимое	Метки	Команды	Комментарии
8004	3E		MVI A,0	Установка А в "0"
8005	00			
8006	32	LOOP:	STA 8020	А → ячейка памяти
8007	20			
8008	80			
8009	00		NOP	
800A	3C		INR A	Прирастить А
800B	C3		JMP LOOP	Переход к LOOP
800C	06			
800Д	80			

Она загружает в аккумулятор нули. Следующая команда - STA 8020 пересылает содержимое аккумулятора в ячейку памяти 8020. Код 32 в ячейке с адресом 8006 указывает, что это команда STA. Когда процессор считывает этот код, он "понимает", что следующие два байта /адреса 8007 и 8008/ содержат адрес, по которому должно быть записано содержимое аккумулятора /в данном случае 8020/. Следует помнить, что байты адреса записываются в обратном порядке. Эта команда не изменяет содержимого аккумулятора, она просто копирует данные в ячейку памяти. За командой STA идет команда NOP, чтобы зарезервировать место для последующего использования. Следующая команда - INP A, она увеличивает содержимое аккумулятора. Когда достигается максимальная величина счета /в двоичном коде - 1111 1111, в шестнадцатеричном - FF или 255 - в десятичном/, содержимое аккумулятора сбрасывается и дальнейший счет идет с нуля. Это обычный режим работы двоичного счетчика.

Последняя команда - переход. Эта команда возвращает программу к команде STA, адрес 8006.

Проведем эксперимент по выполнению описанной программы в шаговом режиме.

1. Введите в память "Микролаб" программу, приведенную в табл.2. Проверьте правильность ее записи в память.

2. Переключите тумблер режима в положение ШАГ.

3. Установите начальный адрес программы /8004/. На двух крайних правых индикаторах появится команда MVI A /код 3E/.

4. Нажмите кнопку ПУСК. Программа начала выполняться в шаговом режиме. Первая команда выполнена. Теперь на двух крайних правых индикаторах высвечено содержимое регистра признаков, а содержимое аккумулятора появилось на третьем и четвертом индикаторах, считая справа. На этих индикаторах видны нули, так как первая команда выполнена и в аккумулятор занесены нули.

5. Нажмите кнопку ПУСК. Выполнена команда STA 8020. Содержимое аккумулятора не изменилось, в ячейке памяти 8020 должны появиться нули. Чтобы проверить содержимое этой ячейки, нажмите кнопки 8, 0, 2, 0, а также кнопку УСТ.АД. На двух крайних правых индикаторах - нули. Это содержимое ячейки 8020. Набирая любой другой адрес, можно просмотреть любую ячейку памяти после каждого шага выполнения программы. Набирая соответствующие адреса, можно также просмотреть и содержимое регистров МП /см.табл.1/.

6. Нажмите кнопку ВОЗВР. Тем самым вы вернулись к выполнению вашей программы и выполнили следующую команду /NOP/. На индикаторах опять появилось содержимое регистра признаков и аккумулятора.

7. Нажмите кнопку ПУСК. Выполнена команда INR A. На индикаторе видна единица, что свидетельствует о приращении содержимого аккумулятора.

8. Проверьте содержимое ячейки 8020. В ней находятся нули, так как перепись единицы из аккумулятора в эту ячейку еще не произошла.

9. Нажав кнопку ВОЗВР., вернитесь к выполнению вашей программы.

10. Нажмите кнопку ПУСК, выполнится следующая команда /STA 8020/, т.е. содержимое аккумулятора переписалось в ячейку 8020.

11. Проверьте содержимое ячейки 8020. Теперь там находится единица.

12. Нажав кнопку ВОЗВР. и далее нажимая кнопку ПУСК, выполните программу по шагам. Проследите, как меняется содержимое аккумулятора.

ОРГАНИЗАЦИЯ ПРОГРАММ

В различных случаях применения перед МП ставятся разные задачи. Например, вольтметр со встроенным МП должен измерить входное напряжение и послать данные на дисплей. Процессор должен перевести данные в вольты и автоматически поддерживать нуль шкалы. Измеритель может иметь клавиатуру /для ввода пользователем специальных запросов/. Для автоматической установки диапазона процессор может быть связан с аттенюатором. Каждая из задач, стоящих перед МП, является в основном независимой и может быть представлена относительно простой программой. Затем одна общая /основная/ программа может объединить все специализированные программы, не вдаваясь в подробности каждой задачи.

Проведем аналогию со структурой большого предприятия. Директор не может выполнить всю работу, но он может принимать все главные решения. Ему не надо беспокоиться о том, какой метлой подметать дворнику, потому что это могут решить другие люди. Существует ряд заместителей, которые отвечают за определенные направления работы. У них в подчинении находятся начальники рангом ниже, которые в свою очередь управляют другими служащими. На предприятии каждый имеет задачу для решения, и чем ниже вы на служебной лестнице, тем больше вы вникаете в детали. Директор имеет время принимать важные решения, потому что он может заставить других людей выполнять менее важные. Они, в свою очередь, заставляют других, которые привлекают еще людей, если это требуется, затем каждый человек докладывает своему начальнику, и наконец, директор получает ответ.

Это очень похоже на работу процессорной системы. Так как задачи могут быть очень различными и сложными, то существует много программ. Одна программа /часто называемая исполнительной/ действует как директор. Программа-исполнитель имеет другие программы, работающие на нее, и которые, в свою очередь, имеют другие программы, работающие на них.

Программы, работающие на другие программы, называются подпрограммами. Подпрограммы могут иметь другие подпрограммы, работающие на них и т.д.

Пример. На рис.3. приведена простая программа, которая обеспечивает запись в ячейку памяти 8020 то нулей, то единиц. В этой программе используются подпрограммы. Одна подпрограмма используется для записи в ячейку памяти единиц, другая - для записи нулей.

Основная программа содержит только три команды: одну - для вызова подпрограммы "нули", другую - для вызова подпрограммы "единицы" и третью - для перехода к началу.

Основная программа определяет все функции программы, но ей не нужно знать адрес ячейки памяти. Потребности определяются подпрограммами.

Команда CALL заставляет МП перейти на подпрограмму. Метки "0" и "1" определяют подпрограммы и помещаются по реальному адресу, когда язык Ассемблер переводится в машинный код. Первые две команды каждой

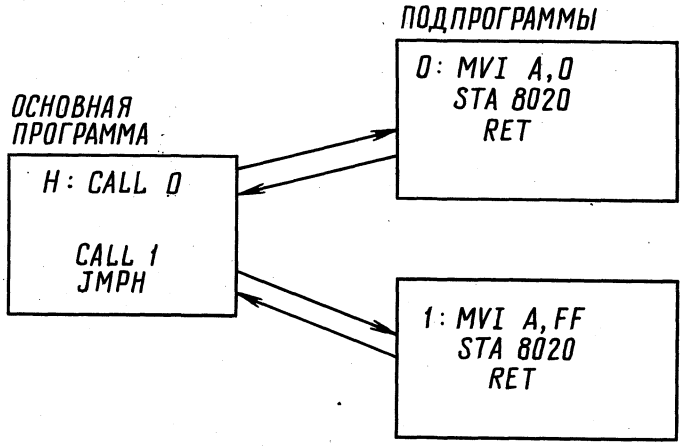


Рис.3. Пример использования подпрограмм

из подпрограмм уже рассмотрены ранее. Последняя команда RET /возврат/ определяет конец подпрограммы и возвращает МП к выполнению основной программы. В табл. 3 показана полная программа с машинными кодами.

Таблица 3

Листинг программы записи в ячейку памяти нулей и единиц

Основная программа

Адрес	Содержимое	Метки	Команды	Комментарии
8000	CD	H	CALL O	Подпрограмма записи нулей
8001	09			
8002	80			
8003	CD		CALL 1	Подпрограмма записи единиц
8004	OF			
8005	80			
8006	C3		JMP H	Повтор
8007	00			
8008	80			

Подпрограмма записи нулей в ячейку памяти 8020

8009	3E	O:	MVI A, 00	Записать в А нули
800A	00			
800B	32		STA 8020	Записать содержимое А в ячейку
800C	20			
800D	80			
800E	C9		RET	Возврат в основную программу

Подпрограмма записи единиц в ячейку памяти 8020

800F	3E	1:	MVI A, FF	Записать в А единицы
8010	FF			
8011	32		STA 8020	Записать содержимое А в ячейку
8012	20			
8013	80			
8014	C9		RET	Возврат в основную программу

CALL состоит из кода операции CD, за которым следует адрес подпрограммы. Адрес подпрограммы хранится таким же образом, как и адрес перехода. Команда RET состоит только из кода C9. Никакого адреса команда не содержит. Однако же МП "знает", куда возвращаться, когда он встречает команду возврата в конце подпрограммы. Когда выполняется команда CALL, адрес возврата, т.е. адрес команды, следующий после вызова подпрограммы, хранится в специальном месте памяти, которое называется СТЕК. Когда в конце подпрограммы встречается команда RET, МП получает из СТЕКА адрес возврата. Программа возвращается к основной программе. Работа СТЕКА осуществляется почти полностью автоматически. Более подробно СТЕК будет описан в последующих разделах.

Проведем эксперимент по использованию программы, приведенной в табл.3.

1. Введите данную программу в память "Микролаб".

2. Поставьте тумблер режима в положение ШАГ.

3. Установите начальный адрес программы /8000/ и нажмите кнопку ПУСК. Выполняется первая команда программы /CALL 0/, и происходит переход к подпрограмме /адрес 8009/.

4. Нажмите кнопку ПУСК. Выполняется следующая команда, которая является первой в подпрограмме. В аккумулятор заносятся нули.

5. Нажмите кнопку ПУСК. Содержимое аккумулятора заносится в ячейку памяти с адресом 8020. Проверьте содержимое этой ячейки. Там записаны нули.

6. Нажмите кнопку ВОЗВР. Выполнится команда RET, и вы вернетесь в основную программу /адрес 8003/.

7. Нажмите кнопку ПУСК. Произойдет переход во вторую подпрограмму /адрес 800F/.

8. Нажмите два раза кнопку ПУСК. В аккумулятор запишутся единицы /FF/, и содержимое аккумулятора перепишется в ячейку памяти с адресом 8020.

9. Проверьте, имеется ли в ячейке 8020 FF.

10. Нажмите кнопку ВОЗВР. Выполнится команда возврата /RET/, и произойдет переход к основной программе /адрес 8006/. По этому адресу будут записаны команды безусловного перехода к адресу 8000.

После этого программа начнет исполняться снова.

11. Нажмите несколько раз кнопку ПУСК, просмотрите программу еще раз.

ПРЕРЫВАНИЯ

Иногда микропроцессорная система должна реагировать на редкие и непредсказуемые события. Вернемся к аналогии с предприятием. Предположим, что могут произойти какие-то экстремальные события. Например, каждый может быть оторван от работы пожарной тревогой. После того, как пожар потушен и его последствия ликвидированы, люди возвращаются к прежней работе.

Аналогичные события могут произойти и в микропроцессорной системе. Например, рассмотрим вольтметр со встроенным МП, который имеет на передней панели кнопку калибровки. Когда нажимается эта кнопка, работа МП прерывается и МП переходит к подпрограмме калибровки. После выполнения калибровки управление возвращается к основной программе, которая была прервана. Эти действия называются прерываниями. Кристалл МП имеет вход, который принимает прерывания. Сигнал высокого уровня на этом входе вызывает переход процессора к специальной подпрограмме прерывания. Эта подпрограмма выполняет задачу, требуемую прерывающим устройством. Команда, стоящая в конце подпрограммы прерывания, возвращает МП к прерванной программе.

Иногда нужно, чтобы МП не реагировал на прерывание. Поэтому прерывание может быть запрещено программой. Существуют две команды: EI - разрешить прерывание, DI - запретить прерывание.

МП КР580ИК80 реагирует на прерывание следующим образом. Не увеличивая содержимого счетчика адреса, он принимает информацию с шины данных, и в зависимости от кода, установленного на шине данных, начинает выполнять программу с определенного адреса в соответствии с табл.4.

Таблица 4

Таблица прерываний

Номер прерывания	Код на шине данных	Адрес, с которого начинается программа обработки прерывания
RST0	C7	0000
RST1	CF	0008
RST2	D7	0010
RST3	DF	0018
RST4	E7	0020
RST5	EF	0028
RST6	F7	0030
RST7	FF	0038

Таким образом, внешнее устройство, вырабатывающее сигнал прерывания, должно выставить на шине данных код, в соответствии с

которым будет выполняться определенная подпрограмма по обработке прерывания. Программист, в свою очередь, должен предусмотреть такую подпрограмму при написании общей программы.

В "Микролаб" прерывание используется для организации шагового режима. Подробнее это будет описано в последующих разделах.

Урок 6

СТРУКТУРА И ФУНКЦИОНИРОВАНИЕ МИКРОПРОЦЕССОРА КР580ИК80

До сих пор мы не рассматривали структуру МП. Однако некоторые знания об устройстве и работе МП будут полезны для более четкого понимания работы всей системы.

УСТРОЙСТВО МИКРОПРОЦЕССОРА КР580ИК80

На рис.1. приведена упрощенная структурная схема МП КР580ИК80. Аккумулятор связан шиной данных с арифметико-логическим устройством /АЛУ/. АЛУ выполняет все действия с данными, например такие, как увеличение числа или сложение двух чисел. Регистр временного хранения соединен с другим входом АЛУ, этот регистр недоступен программисту. Он автоматически управляется схемами управления МП. Флаги представляют собой набор триггеров, которые показывают результат последней операции, выполненной АЛУ. Например, флаг нуля устанавливается, если результат операции равен нулю.

Регистр кода операции, дешифратор кода операции, программный счетчик и устройство управления и синхронизации используются для выбора команд из памяти и их выполнения. Предположим, что должна быть считана и выполнена команда, хранящаяся в ячейке с адресом 8000. Программный счетчик через шину адреса выдаст на устройство памяти адрес 8000. Из памяти на шину данных поступит содержимое ячейки 8000 /код операции/, и МП запишет его в регистр кода операции. Этот регистр передаст информацию дешифратору кода

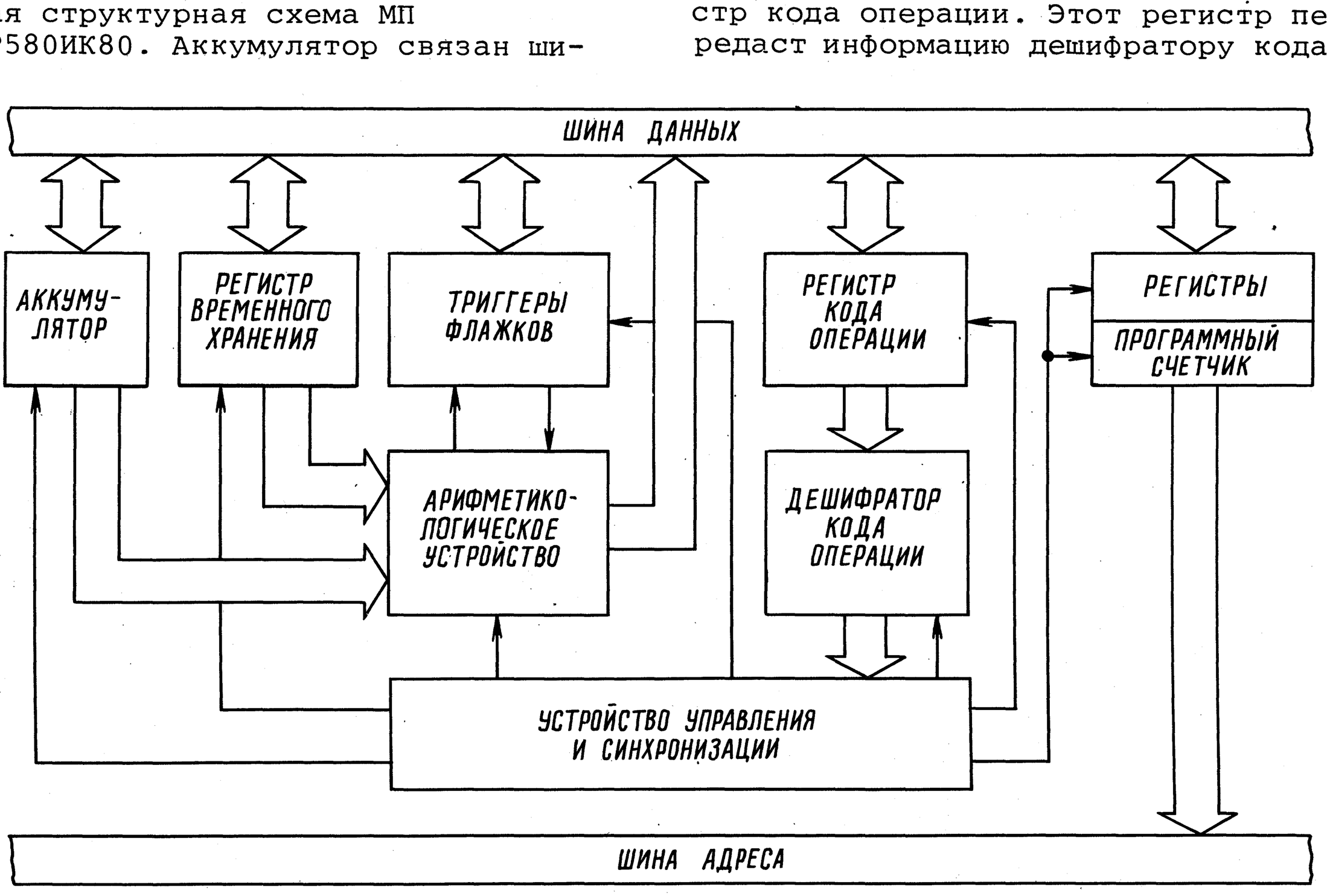


Рис.1. Структурная схема микропроцессора КР580ИК80

операции, который распознает код и выдает управляющие сигналы для схем синхронизации и управления. Схемы управления и синхронизации - это как бы процессор внутри процессора. ПЗУ внутри МП содержит микрокод /или микропрограмму/, который точно указывает процессору, что делать, чтобы выполнить каждую команду. Записанный микрокод, который обычно задается изготовителем МП, называемый микропрограммой, определяет язык МП и не может изменяться при записи программ, выполняемых процессором.

Например, для команды MVI A в устройство управления и синхронизации поступает дешифрованный код операции 3E. Затем увеличивается адрес в программном счетчике. Устройство управления определяет,что за этим кодом операции следует байт данных, так что содержимое ячейки памяти, указанной программным счетчиком, считывается в аккумулятор. Микропрограмма укажет управляющей логике, что команда выполнена. Содержимое программного счетчика увеличивается, и следующий байт программы /следующий код/ считывается в регистр кода операции. Затем начнется исполнение команды.

ОСНОВНЫЕ ОПЕРАЦИИ МИКРОПРОЦЕССОРА

МП КР580ИК80 может выполнять четыре основных типа операций:

1 - чтение данных из памяти или порта ввода;

2 - запись данных в память или порт вывода;

3 - выполнение внутренних операций МП.

4 - передача управления другой ячейке памяти.

Первые два типа операций не требуют объяснений. Третий /внутренние операции/ включает действия с регистрами /в том числе с аккумулятором/ без обращения к памяти или портам ввода/вывода. Например, содержимое одного регистра может быть переписано в другой регистр или содержимое регистра может быть увеличено или уменьшено. Четвертая группа включает в себя такие команды, как JMP, CALL, RET.

СИНХРОНИЗАЦИЯ МИКРОПРОЦЕССОРА И ИНФОРМАЦИЯ О СОСТОЯНИИ

Командный цикл определяется как промежуток времени, необходимый для выборки и выполнения команды. Формат команд МП КР580ИК80 содержит от одного до трех байтов. Каждый командный цикл состоит из одного, двух, трех, четырех или пяти машинных циклов. Машинный цикл требуется каждый раз, когда центральный процессор обращается к памяти или портам ввода/вывода. Этап выборки в командном цикле требует одного машинного цикла для каждого выбираемого байта. Длительность выполнения командного цикла зависит от вида выбранной команды. Некоторые команды не требуют дополнительных машинных циклов, кроме тех, которые необходимы для выборки команды. Однако другие команды требуют дополнительного машинного цикла для записи и считывания данных в память или из памяти и в порт вывода или из порта ввода. Каждый машинный цикл состоит из трех, четырех или пяти тактов. Длительность каждого такта равна длительности одного периода синхросигнала /длительность такта при частоте 2 МГц - 0,5 мкс/. Логическая схема синхронизации МП требует два несовпадающих тактовых импульса Ф1 и Ф2 /рис.2/. Импульс Ф1 является тактовым импульсом, делящим машинный цикл на такты.

В начале каждого машинного цикла МП вырабатывает сигнал СИНХР. В течение действия этого сигнала на шину данных выдается байт состояния /информация о состоянии/. Этот байт расшифровывается внешними устройствами. В нем содержится информация о том, что будет происходить в данном машинном цикле:

запись в память, чтение из памяти, запись в порт, чтение из порта.

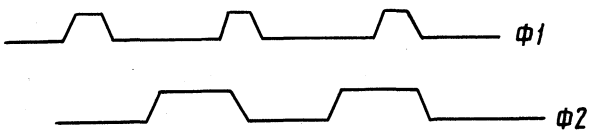


Рис.2. Тактовые импульсы для синхронизации работы МП

ВЫПОЛНЕНИЕ ПРОГРАММЫ

В общем случае МП последовательно считывает одну ячейку памяти за другой, выполняя указанные операции. Исключением из этого правила наблюдается при выполнении команд перехода, вызова или возврата, а также при прерывании выполняемой программы. МП прерывает последовательное считывание и начинает выполнение команд с другого адреса.

Заметим, что коды операций и данные перемешаны в памяти. Один адрес может содержать код, следующие два - адрес перехода, затем опять код и снова данные. Программист должен быть уверен в том, что память содержит правильную последовательность кодов и данных. МП не может обнаружить различие между ними. Коды, адреса переходов и данные - это просто комбинации битов, хранимых в памяти. Вся эта информация считывается одним и тем же способом и передается по одной и той же шине данных. МП всегда должен иметь информцию о том, считывается код операции или данные, и обрабатывать их соответственно. Процессор исходит из того, что первая ячейка, которую он считывает, содержит код операции. Если код операции требует байта данных, то МП "знает", что следующий байт - данные, и обрабатывает их соответствующим образом. Байт, следующий за байтом данных, МП воспринимает как код следующей операции. Если данные будут неправильно истолкованы и обработаны как код операции, то выполнение программы полностью выйдет из-под контроля.

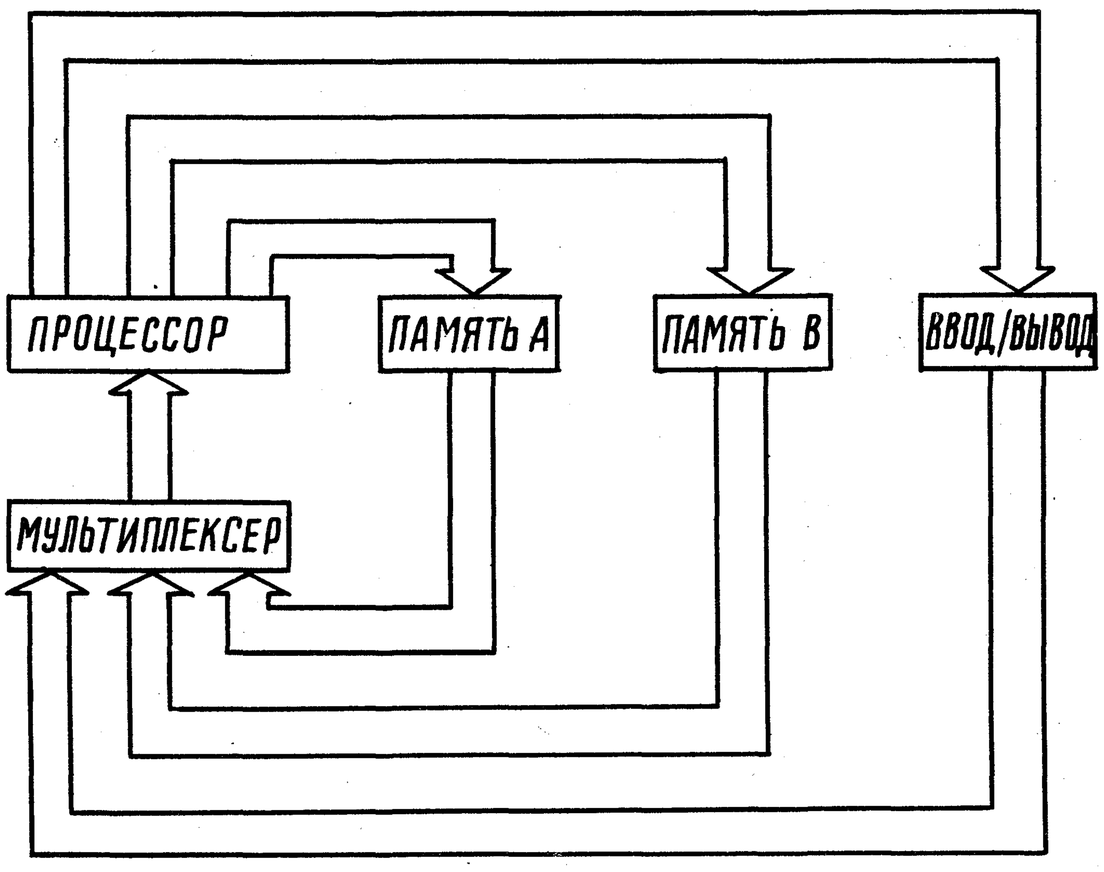


Рис.1. Обмен данными с использованием традиционных методов обработки

АППАРАТНЫЕ СРЕДСТВА

Урок 7

ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ АППАРАТНОЙ ЧАСТИ

Микропроцессорные системы выполняются на основе шин, которых нет в традиционных логических устройствах схемной логики. В микропроцессорной системе между многими устройствами и процессором происходит обмен данными /рис.1/. Процессор должен иметь набор выходов данных для каждого устройства и мультиплексер для выбора устройства ввода данных. По шине данных обычно передают 8 битов данных. Поэтому для простой системы, состоящей из трех устройств, требуется 48 линий: 24 - для ввода, 24 - для вывода данных. Более сложная система должна иметь несколько ЗУ и шин ввода/вывода и сотни межсоединительных линий.

На рис.2. приведена упрощенная схема межсоединительных линий. Один комплект соединений используется для соединения МП со всеми устройствами ввода/вывода, входящими в систему. Такая структура может расширяться лишь при небольшом усложнении межсоединений. При использовании шин все устройства хотя и связываются одной и той же линией данных, но только одно устройство может выдавать данные в определенное время. Линии адреса и управления обеспечивают выбор определенного устройства.

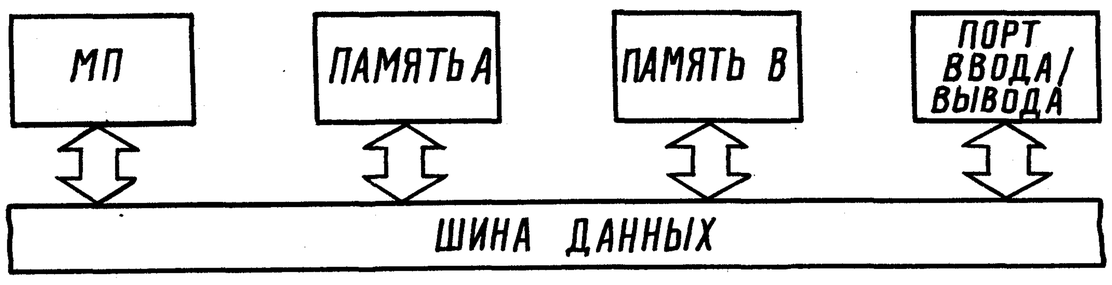


Рис.2. Обмен данными, использующий шину для уменьшения количества межсоединительных линий

ШИНА ТРЕХ СОСТОЯНИЙ

Драйвер трех состояний дает возможность применять шину с разделенными данными. Этот принцип остается неизменным независимо от числа линий в шине. Типичная шина данных имеет восемь линий.Шина трех состояний подобна телефонной линии. К такой шине можно подключать много передатчиков и приемников. На рис.3 показана шина цифровой схемы с четырьмя передатчиками /драйверами трех состояний/ и двумя приемниками /обычными вентилями/. Драйверы трех состояний имеют выход на общую шину, но информация на нее поступает только от одного передатчика, на который подан разрешающий сигнал управляющей логики. В этот момент информация остальных драйверов не имеет значения, так как их выходы находятся в высокоимпедансном состоянии и, следовательно, будут изолированы от шины.

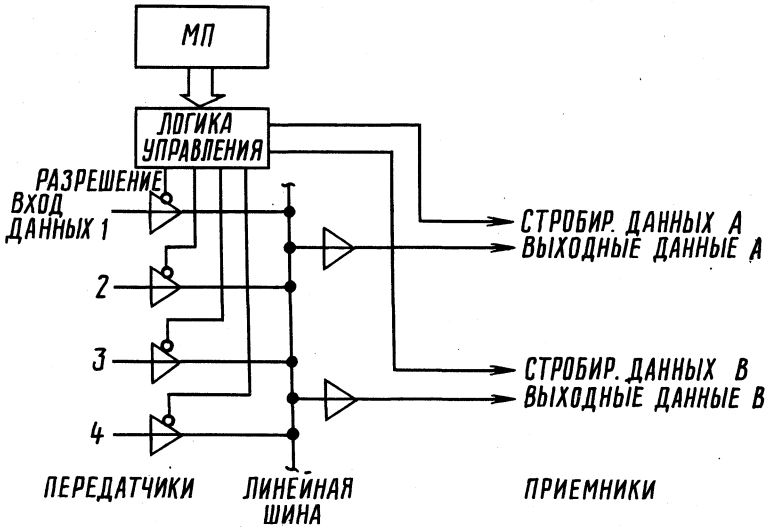


Рис.3. Однолинейная шина, имеющая три состояния с четырьмя передатчиками и двумя приемниками

Аналогично происходит прием информации с шины. Независимо от числа приемников, информацию принимает тот, которому дан разрешающий сигнал управляющей логики. Для выработки разрешающих сигналов управляющей логики, приходящих из МП, используют адресные и управляющие шины /рис.4/.

Описанные устройства являются однонаправленными. Это либо приемники, либо передатчики. Существуют

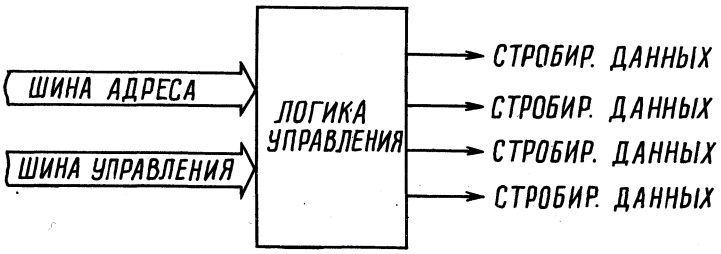


Рис.4. Схема выбора управляющей логикой устройства переноса данных

также двунаправленные устройства, являющиеся и приемниками, и передатчиками. На рис.5 показана шина с двумя приемниками-передатчиками. Здесь показаны только два устройства но их может быть и больше. Каждый приемник-передатчик имеет два управляющих сигнала: сигнал разрешения выхода для драйвера трех состояний и стробирования данных для входа. Примером двунаправленного устройства является ОЗУ.

Рассмотрим рис.5. Устройство А должно переслать данные на устройство В. Управляющая логика устанавливает разрешение выхода А в истинное /разрешенное/ состояние и запрещение выхода В /не разрешенное/. После того, как прошло время, достаточное для того, чтобы данные достигли входа данных устройства В, контроллер посылает импульс по линии В стробирования данных. Это заставляет устройство В считывать данные с шины которые были переданы устройством А. Следует отметить, что множество других устройств может подсоединяться к этой шине, однако пока нет сигналов разрешения, они не будут принимать данные.

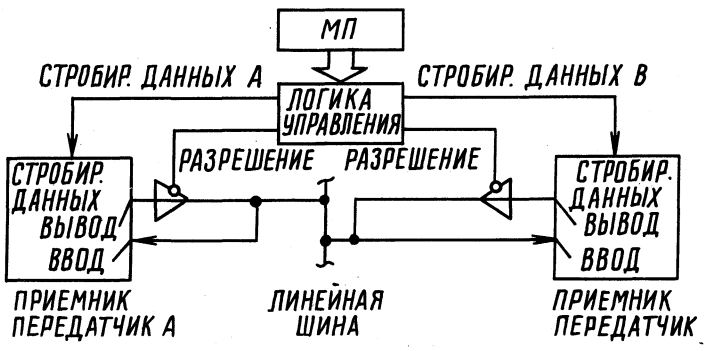


Рис.5. Двунаправленные передатчики-приемники, подсоединенные к линии шины

ШИНА ДАННЫХ

Шина данных "Микролаб" является двунаправленной 8-разрядной шиной трех состояний. МП и ОЗУ являются и приемниками, и передатчиками. ПЗУ является только передатчиком, порт ввода/вывода - и приемником, и передатчиком.

На рис.6 показано, как эти устройства соединены с шиной данных. МП ОЗУ, ПЗУ и порт ввода/вывода имеют на выходах драйверы трех состояний. Входы выбора кристалла дают разрешение драйверам и вызывают появление данных из выбранного устройства на шине данных.

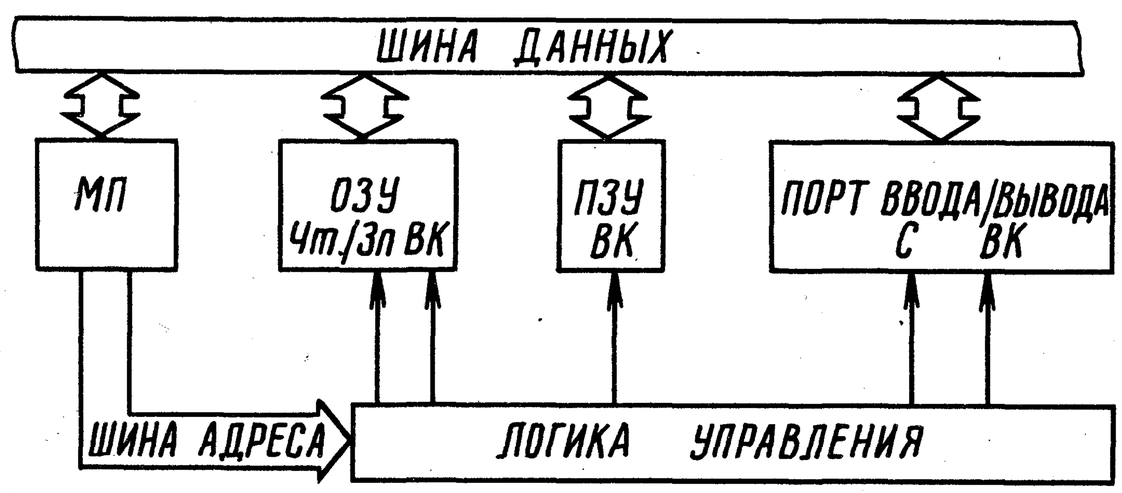


Рис.6. Выходы устройств с тремя состояниями соединяются с МП по шине данных

МП в системе действует как диспетчер, он гарантирует, что в определенный момент шину использует только одно устройство. Если МП должен считать данные с ПЗУ, то он сначала запрещает свои собственные выходы данных, а затем генерирует управляющие сигналы, требующиеся для подключения ПЗУ к шине данных. Затем информация с выходов ПЗУ появляется на шине данных, и МП считывает данные. Считывание информации из ОЗУ и порта ввода/вывода выполняется подобным образом. Чтобы записывать данные в устройство, такое как ОЗУ или порт ввода/вывода, МП сначала помещает записываемые данные на шину данных. Затем он генерирует управляющие сигналы, которые посылают импульс записи на соответствующие устройства. Импульс затем заставляет устройство записать у себя эти данные.

Данные идут через МП. Например, чтобы передать данные из порта ввода/вывода в ОЗУ, МП сначала считывает данные с порта, а затем записывает их в ОЗУ. Из-за того, что данные не могут передаваться прямо с порта ввода/вывода в ОЗУ, они должны временно запоминаться в МП.

Таким образом, шина данных используется для переноса данных в системе. Все устройства используют одну и ту же шину. Управляющая логика, работающая от сигналов, генерируемых МП, командует устройству, когда оно должно помещать данные на шину или считывать данные с шины.

В микролаборатории шина данных имеет восемь линий. МП КР580ИК80 может одновременно обрабатывать 8 битов данных и называется 8-разрядным МП. Ранние МП использовали 4-разрядную шину, а некоторые новые устройства используют 16-разрядную шину данных.

ШИНА АДРЕСА

Мы рассмотрели, как шина данных используется многими устройствами для обмена данными. Теперь нужно определить метод, с помощью которого МП мог бы выбрать определенное устройство, соединяющееся с шиной данных. Шина адреса вместе с управляющей шиной выполняет эту функцию. Благодаря тому, что шина адреса является однонаправленной, ее работа гораздо проще, чем работа шины данных. Каждое устройство, хранящее данные, имеет определенный адрес. Перед выполнением любого переноса данных /через шину данных/ МП должен выдать адрес устройства, для которого предназначены данные.

Шина адреса МП КР580ИК80 имеет 16 линий, что обеспечивает прямую адресацию 216 - 65536 ячеек памяти. Эти линии обозначаются А0, А1...А15. А0 - самый младщий бит.

ДЕШИФРАТОР АДРЕСА

Дешифратор адреса является частью управляющей логики, он генерирует сигналы выбора устройства, когда на шине адреса присутствуют определенные адреса.

Например, на рис.7 показан адресный дешифратор для адреса 3000 /0011 0000 0000 0000 - двоичный код/.

Выход этого дешифратора является истинным /лог. 0/, когда точно этот адрес присутствует на адресной шине.

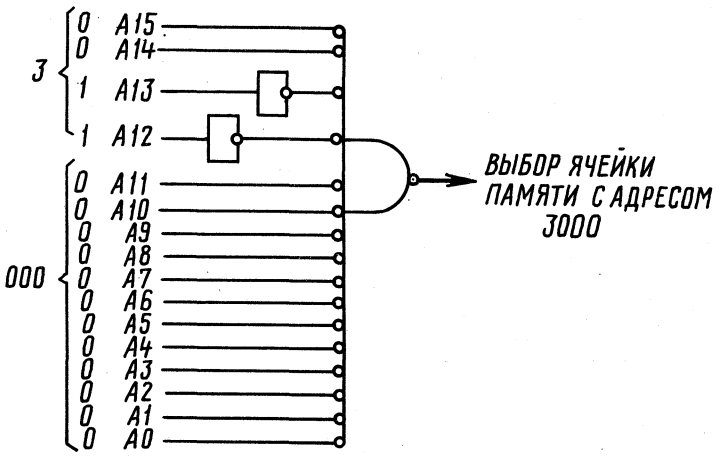


Рис.7. Выбор ячейки памяти с адресом 3000

УПРАВЛЯЮЩАЯ ШИНА

Весь процесс координируется управляющей шиной. Большинство управляющих сигналов генерируется МП.

Сигналы ЧТ и ЗП - это основные управляющие сигналы МП КР580ИК80. Если ЧТ находится на активном уровне /лог. 1/, то выполняется операция считывания, МП сигнализирует, что он считывает данные с шины данных. Если сигнал ЗП находится в состоянии лог. 0, то это означает, что выполняется операция записи. МП помещает данные на шину данных и сигнализирует адресуемому устройству запомнить эти данные. Основное различие между управляющей шиной и шинами адреса и данных заключается в том, что каждый провод в управляющей шине выполняет отдельную функцию, в то время как у шин адреса и данных каждая линия передает одинаковый тип информации.

ПОРТ ВВОДА/ВЫВОДА

Порт ввода/вывода в "Микролаб" построен на микросхеме КР580ИК55. Микросхема представляет собой однокристальное программируемое устройство ввода/вывода параллельной информации. Этот программируемый периферийный интерфейс /ППИ/ предназначен для сопряжения МП с различными периферийными устройствами. Для этого в ППИ имеется три 8-разрядных порта ввода/вывода /рис.8/.

Каждый порт может быть запрограммирован либо на ввод, либо на

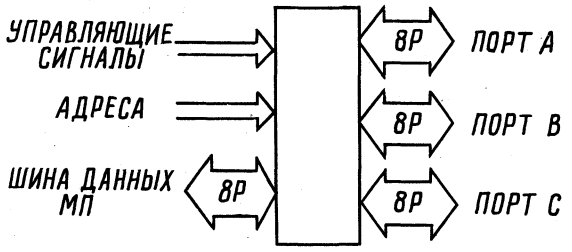


Рис.8. Программируемый периферийный интерфейс

вывод информации, т.е. сначала на ППИ подается управляющее слово, которое его программирует, а потом с помощью шины адреса и управляющих сигналов происходит выбор того или иного порта данного интерфейса.

ДЕШИФРАЦИЯ АДРЕСА ДЛЯ ПЗУ

Предположим, что дешифрация адреса требуется для ПЗУ, содержащего 512 байтов /рис.9/. Поскольку шина адреса содержит 16 разрядов /А0-А15/, а для дешифрации адреса ПЗУ требуется только девять младших /А0-А8/, то оставшиеся старшие разряды А9-А15, инвертированные и объединенные по схеме "И" с сигналом ЧТЕНИЕ, используются для разрешения выбора ПЗУ. Информация разрядов А0-А8 дешифрируется внутренним дешифратором самого ПЗУ, а для дешифрации разрядов А9-А15 требуется отдельный внешний дешифратор. Однако большинство микропроцессорных систем используют более одного кристалла ЗУ, т.е. нужен более сложный адресный дешифратор. Так например, в "Микролаб" три вышеописанных 512-байтных ПЗУ соединены последовательно и соответственно имеют адреса, приведенные в табл.1.

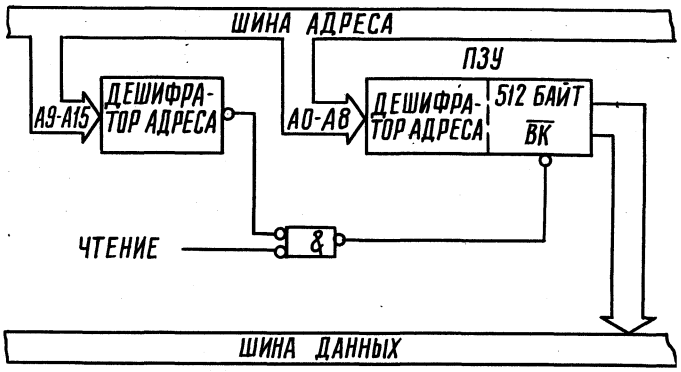


Рис.9. Дешифрация адреса для ПЗУ

Таблица 1

Адреса, назначенные каждому из трех 512-байтных ПЗУ "Микролаб"

ПЗУ	Адреса ПЗУ	Разряды адреса 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ПЗУ 0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	.	
	.	
	.	
	511	0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
ПЗУ 1	512	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
	513	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
	.	
	.	
	.	
	1023	0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
ПЗУ 2	1024	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
	1025	0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1
	.	
	.	
	.	
	1535	0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1

Теперь адресные линии должны указывать не только, какое слово в кристалле будет адресоваться, но и какой кристалл памяти будет выбираться.

Младшие 9 битов адреса указывают ячейку в каждом кристалле, а старшие 7 битов указывают адресуемый кристалл.

Отметим, что только биты A9 и A10 отличаются друг от друга при дешифрации одного кристалла. На рис.10 показано, как выполняется рассмотренная адресация.

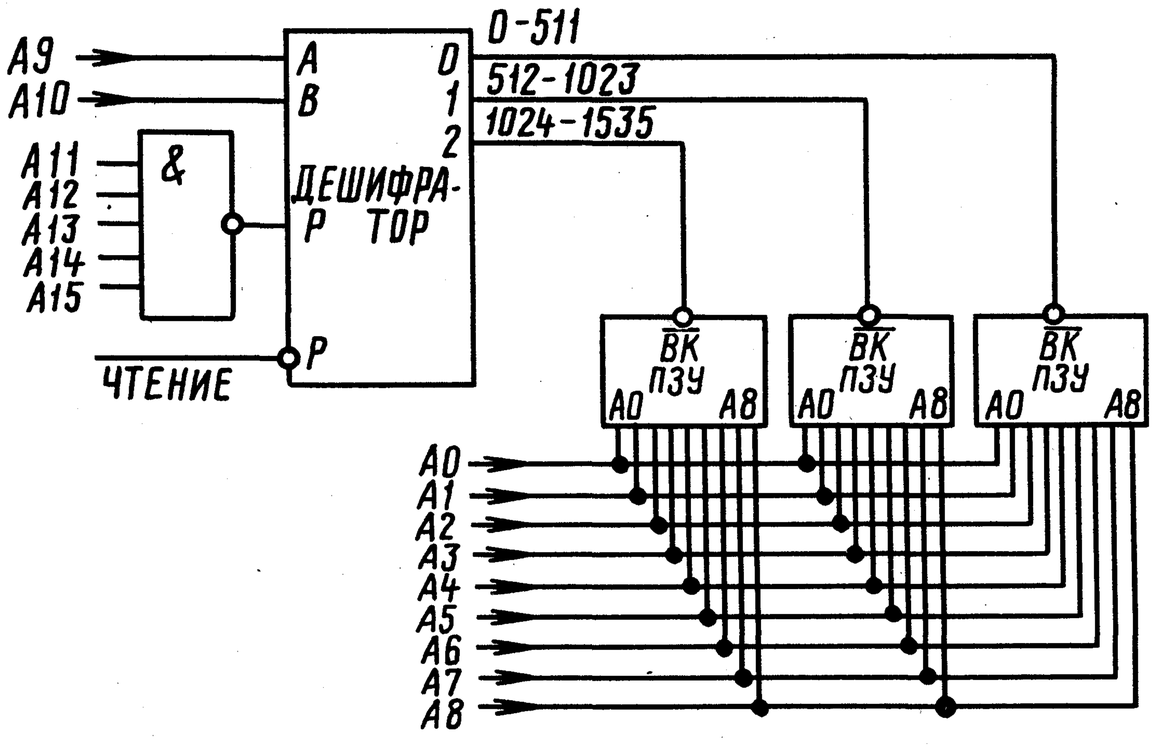


Рис.10. Дешифрация адреса для выбора кристаллов ПЗУ

Младшие 9 битов адреса непосредственно поступают на адресные линии всех трех ПЗУ. Это объясняется тем, что эти биты указывают ячейку внутри кристалла. Затем дешифратор адреса "рассматривает" 7 старших битов адреса и генерирует сигналы ВК. Два младших бита старшей половины A9 и A10 используются для выбора кристаллов. Оставшиеся старшие биты используются для разрешения дешифратора только тогда, когда они все находятся в низком состоянии. Следует заметить, что управляющий сигнал ЧТЕНИЕ используется как сигнал разрешения дешифратора. Это эквивалентно объединению по схеме "И" этого сигнала с каждым выходом дешифратора.

Принцип дешифрации в "Микролаб" аналогичен описанному. Однако дешифратор выполнен на других логических элементах и поэтому выглядит несколько сложнее.

УПРАВЛЕНИЕ ОЗУ

ОЗУ дешифрируется так же, как ПЗУ, за исключением некоторых дополнительных управляющих схем записи

и считывания. Кроме входа ВК, ОЗУ имеет вход ЗАПИСЬ /табл.2/.

Таблица 2

Таблица истинности ОЗУ

ВК	ЗАПИСЬ	Функция
0	0	Запись
0	1	Чтение
1	0	Кристалл не выбран
1	1	

ВК должен быть в низком состоянии для того, чтобы выполнилось считывание или запись. Если ЗАПИСЬ находится в высоком состоянии, а ВК находится в низком, ОЗУ выдает данные на шину данных, чтобы процессор мог их считывать. Если оба сигнала ЗАПИСЬ и ВК находятся в низком состоянии, то в ОЗУ записываются данные, находящиеся на шине данных. Эти данные записываются в ту ячейку ОЗУ, на которую в данный момент указывает шина адреса. Схема, предназначенная для выполнения нужного стробирования, показана на рис.11.

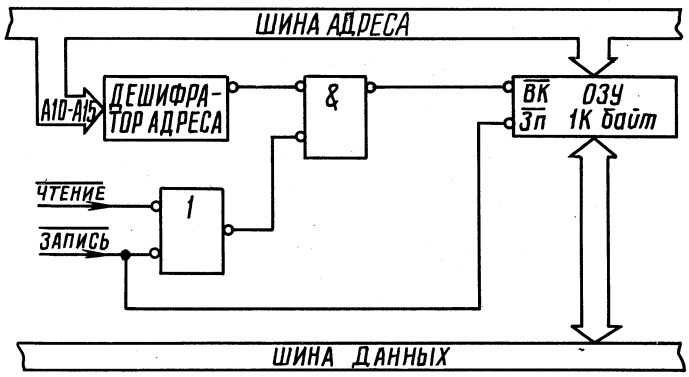


Рис.11. Дешифрация адреса и управления для 1К-байтного ОЗУ

В "Микролаб" используются кристаллы ОЗУ 1Кх1, т.е. для того, чтобы организовать память 1Кх8, необходимо восемь кристаллов, соединенных параллельно, как это и реализовано в "Микролаб". Адресные и управляющие сигналы поступают параллельно на все восемь кристаллов ОЗУ, а линии данных этих кристаллов образуют 8-разрядную шину данных.

Урок 8

ДЕШИФРАЦИЯ АДРЕСА

СТРУКТУРА АДРЕСАЦИИ МИКРОПРОЦЕССОРНОЙ ЛАБОРАТОРИИ

Микропроцессорная лаборатория КР580ИК80 включает в себя блок памяти, который состоит из ОЗУ и ПЗУ. В табл.1 показана карта памяти микро-ЭВМ.

В основном объеме памяти монитору отводятся адреса с 0000 по 02FF. Но можно расширить эту область до 03FF, запрограммировав соответствующее ППЗУ.

Основному ОЗУ емкостью 1К байт отведены адреса с 8000 по 83FF, в которых старшие 57 байт отводятся рабочей области монитора, а остальные могут использоваться по выбору пользователя.

Другие адреса, кроме определенных выше, не используются и остаются в распоряжении пользователя, так как все шины адреса и данных выводятся на разъем платы микро-ЭВМ для стыковки с внешним устройством.

АППАРАТНАЯ ЧАСТЬ СХЕМЫ ДЕШИФРАЦИИ

Рассмотрим принцип работы схемы дешифрации адреса, состоящей из дешифратора /К155ИД4/ и логических схем /рис.1/. Принципиальная электрическая схема приведена в приложении.

Если на входах разрешения дешифратора D27 /ножки 2 и 14/ появляется сигнал лог. 0, то дешифратор

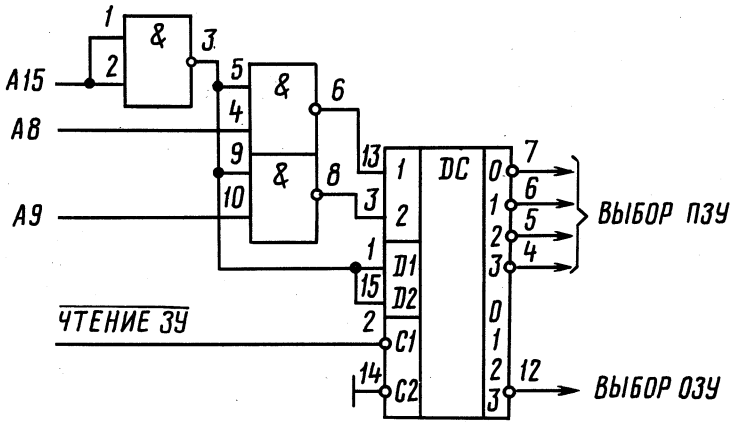


Рис.1. Схема дешифрации адреса МЛ

Таблица 1

Карта памяти микро-ЭВМ

Адреса в двоичном коде по разрядам 15 14 13 12 11 10 9 8	в шестнадцатеричном коде	Емкость памяти	ПЗУ/ОЗУ	Использование
1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0	FFFF | 8400	31К		Неиспользуемая область
1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0	83FF | 8000	1К	ОЗУ	Рабочая область монитора Область пользователя
0 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0	7FFF | 0400	31К		Неиспользуемая область
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1	03FF | 0300	256	ППЗУ	Дополнительная область монитора
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	02FF | 0000	768	ППЗУ	Область монитора

открывается. Тогда на всех выходах дешифратора появляются лог.1, кроме одного лог.0 в зависимости от комбинации на адресных входах /13,3, 1, 15/ дешифратора. Этот нулевой сигнал и разрешает выбор нужной ячейки ОЗУ или ПЗУ в соответствии с сигналами, которые разрешают чтение/запись ОЗУ, чтение ПЗУ. С приходом уровня лог.0 на входы УВ микросхем D48 /15/, D49 /15/ разрешается вывод информации из ОЗУ на шину данных.

Для согласования ОЗУ с двунаправленной шиной данных используется двунаправленный шинный формирователь /микросхемы D48, D49/. В случае отсутствия лог.0 на входе УВ шинные формирователи либо находятся в режиме запрета, тогда их выходные значения находятся в третьем состоянии, либо в режиме ввода информации в ОЗУ, тогда производится запись данных.

Выбор этого режима осуществляется в соответствии с сигналом ЗАПИСЬ, поступающим с процессора. На входы ОЗУ /ножка 3/ приходит лог.0, разрешающий запись в ОЗУ /микросхемы D33 - D36, D38 - D41/.

За выбор ячейки ОЗУ или ПЗУ отвечает старший разряд адресной шины А15. ВК из группы ПЗУ в конечном счете выполняется адресным разрядом А9, поступающим на дешифратор в соответствии с табл.2.

Таблица 2

Адресация кристаллов ПЗУ

A15	A9	D45	D46
0	0	1	0
0	1	0	1

НЕКОТОРЫЕ СПОСОБЫ ДЕШИФРАЦИИ

Для выполнения дешифрации адреса можно использовать другие способы. Выбор способа зависит от множества факторов, таких как объем памяти, число периферийных устройств, типы используемых устройств ввода/вывода и ЗУ.Некоторые устройства имеют разрешающие выводы, используемые при дешифрации адреса.

ДЕШИФРАЦИЯ ЛИНЕЙНОГО ВЫБОРА

Линейный выбор, самый простой способ дешифрации, не использует логику дешифрации адреса. Старшие разряды шины адреса действуют непосредственно как сигналы выбора кристалла. На рис.2 показан пример дешифрации линейного выбора, ОЗУ выбирается каждый раз, когда A15 находится в высоком состоянии /лог.1/.

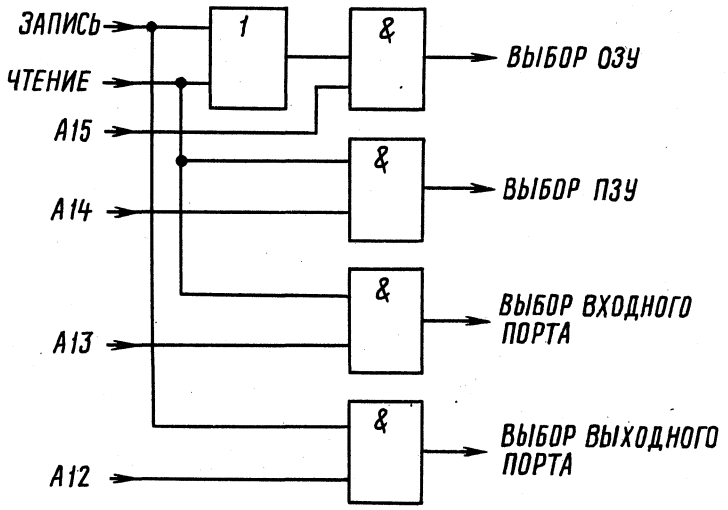


Рис.2. Дешифрация линейного выбора

Это соответствует всем адресам от 8000 до FFFF. ПЗУ выбирается каждый раз, когда A14 находится в высоком состоянии, что соответствует адресам 4000-7FFF. Однако ПЗУ также выбирается, если A14 и A15 находятся в высоком состоянии, соответствующем адресам C000-FFFF. Это перекрывает адресное пространство ОЗУ. На оба устройства поступает сигнал разрешения, если делается попытка считать информацию по одному из этих адресов.

Это вызывает сбой шины, что является недостатком схемы линейной дешифрации. Другим недостатком этого способа является потеря большого адресного пространства. Поэтому этот метод ограничивается небольшими системами дешифрации.

ДЕШИФРАЦИЯ С ПОМОЩЬЮ ЛОГИЧЕСКОГО КОМПАРАТОРА

Наиболее простой и гибкий способ дешифрации - дешифрация с помощью логического компаратора. При этом выбирается одна часть из 2^N возможных адресных полей из N адресных входов. На рис.3 показана схема, генерирующая один сигнал выбора устройства и шесть старших адресных битов системы. Каждый вход A компаратора сравнивается с соответствующим входом B. Когда все 6 пар входов совпадут, выход компаратора переходит в низкое состояние. Для установки логического уровня на входах B компаратора используются переключа-

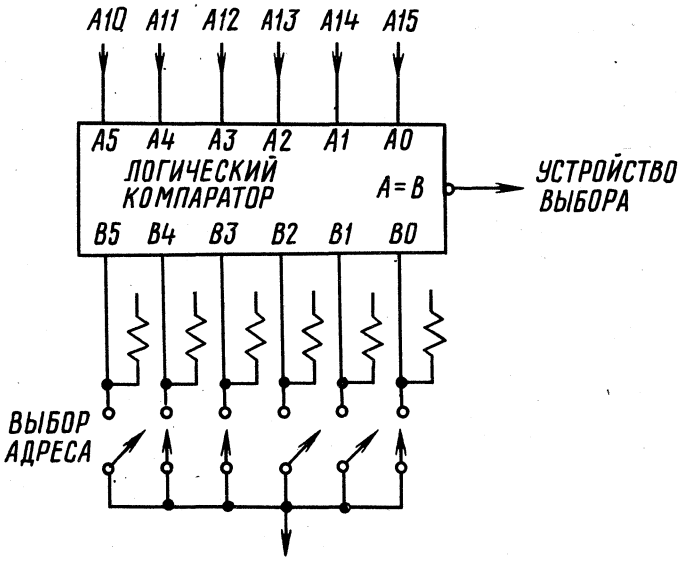


Рис.3. Логический дешифратор компаратора

тели. Этот способ удобен для дешифрации ЗУ периферийных плат, где находятся переключатели или перемычки, устанавливающие адрес каждой платы в системе.

Вентили, выполняющие функцию "исключающего ИЛИ", также используются для выполнения функции компаратора.

ДЕШИФРАЦИЯ С ПОМОЩЬЮ КОМБИНАЦИОННОЙ ЛОГИКИ

При этом способе дешифрации используются стандартные логичес-

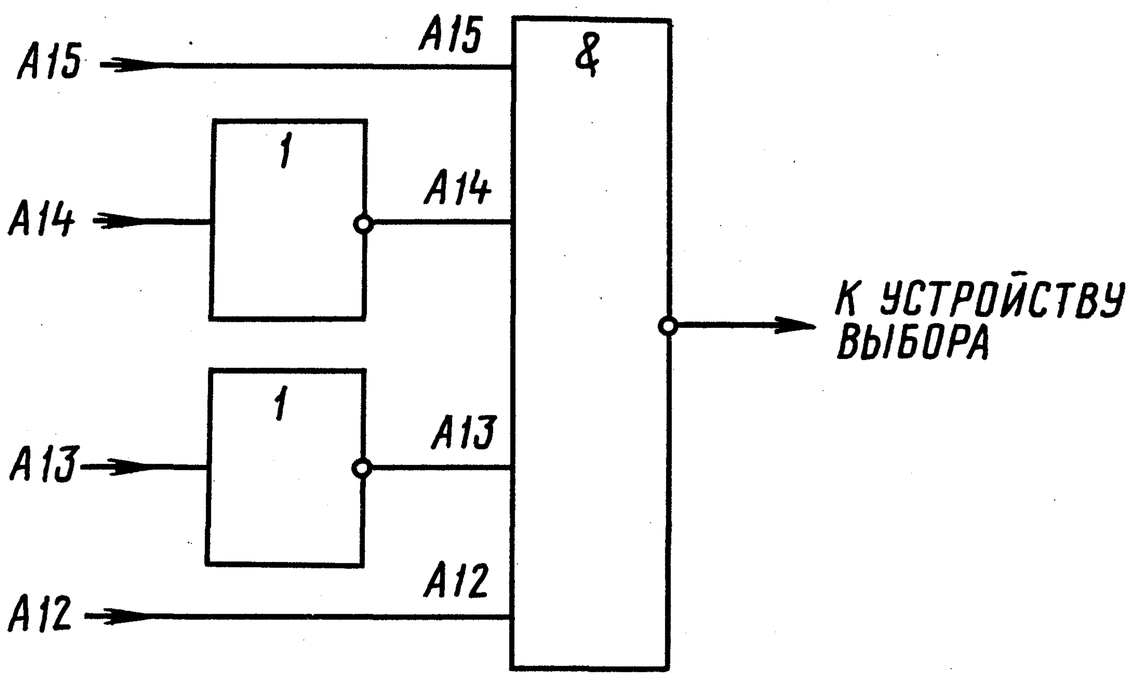


Рис.4. Дешифратор на логических вентилях

кие вентили. На рис.4 показан 4-входовый вентиль И-НЕ, которому предшествуют инверторы, дешифрирующие адреса с 9000 до 9FFF. Выход переходит в низкое состояние каждый раз, когда адресные линии A15-A12 находятся в состоянии 1001.

Урок 9

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Имеются два типа полупроводниковых ЗУ: ОЗУ и ПЗУ. Каждый из них имеет множество разновидностей. В данном разделе описываются некоторые наиболее широко используемые устройства.

ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

ОЗУ используются в программируемой части памяти пользователя для запоминания данных почти во всех системах с микро-ЭВМ. Существует два типа ОЗУ: статические и динамические.

Статические ОЗУ используют триггер для каждого элемента памяти, поэтому ОЗУ с объемом в 1К имеют 1024 триггера. Каждый триггер может устанавливаться для запоминания лог.1 и сбрасываться для запоминания лог.0. Схемы дешифрации адреса внутри кристалла ОЗУ выбирают определенный триггер, указываемый адресными линиями. Состояние триггера не изменится до тех пор, пока он не будет заполнен новыми данными или же не прервется питание ОЗУ.

В динамических ОЗУ для хранения информации используют конденсатор. В конденсаторе запоминается заряд, индицирующий лог.1, отсутствие заряда индицирует лог.0. Этот метод проще, однако конденсатор может потерять свой заряд, и через несколько микросекунд лог.1 может стать лог.0. Поэтому заряды должны регенерироваться. Регенерирование заключается в считывании последовательности адресных ячеек ОЗУ за определенное время. В процессе считывания данных кристалл ОЗУ автоматически переписывает ту же самую информацию в считываемую ячейку. В результате все лог.1 восстанавливаются до полного заряда, а все лог.0 - до полного отсутствия заряда. Динамические ЗУ обычно регенерируют, по крайней мере, каждые 2 мкс.

Из-за необходимости постоянного восстановления информации системы с динамическими ЗУ должны усложняться дополнительными специальными схемами. Но динамические ЗУ более дешевы и потребляют меньшую мощность по сравнению со статическим ЗУ такого же объема. Поэтому, статические ЗУ обычно используются лишь в небольших системах, тогда как большие ОЗУ строятся на динамических ЗУ.

ОЗУ микропроцессорной лаборатории построено с использованием статического ЗУ емкостью 1К бит. Оно построено на МДП БИС ЗУ К565РУ2 с организацией 1Кх1 битов. Чтобы получить 8-разрядное слово, надо взять 8 таких ЗУ. На рис.1 показана схема соединения кристаллов ОЗУ в "Микролаб".

ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

В ПЗУ хранится постоянная информация. Так например, программа монитора микролаборатории запоминается в ПЗУ, ее всегда можно оттуда считать и выполнить. ОЗУ из-за того, что они теряют свое содержимое при отключении питания, не подходят для запоминания постоянных программ.

Существуют четыре различных типа ПЗУ.

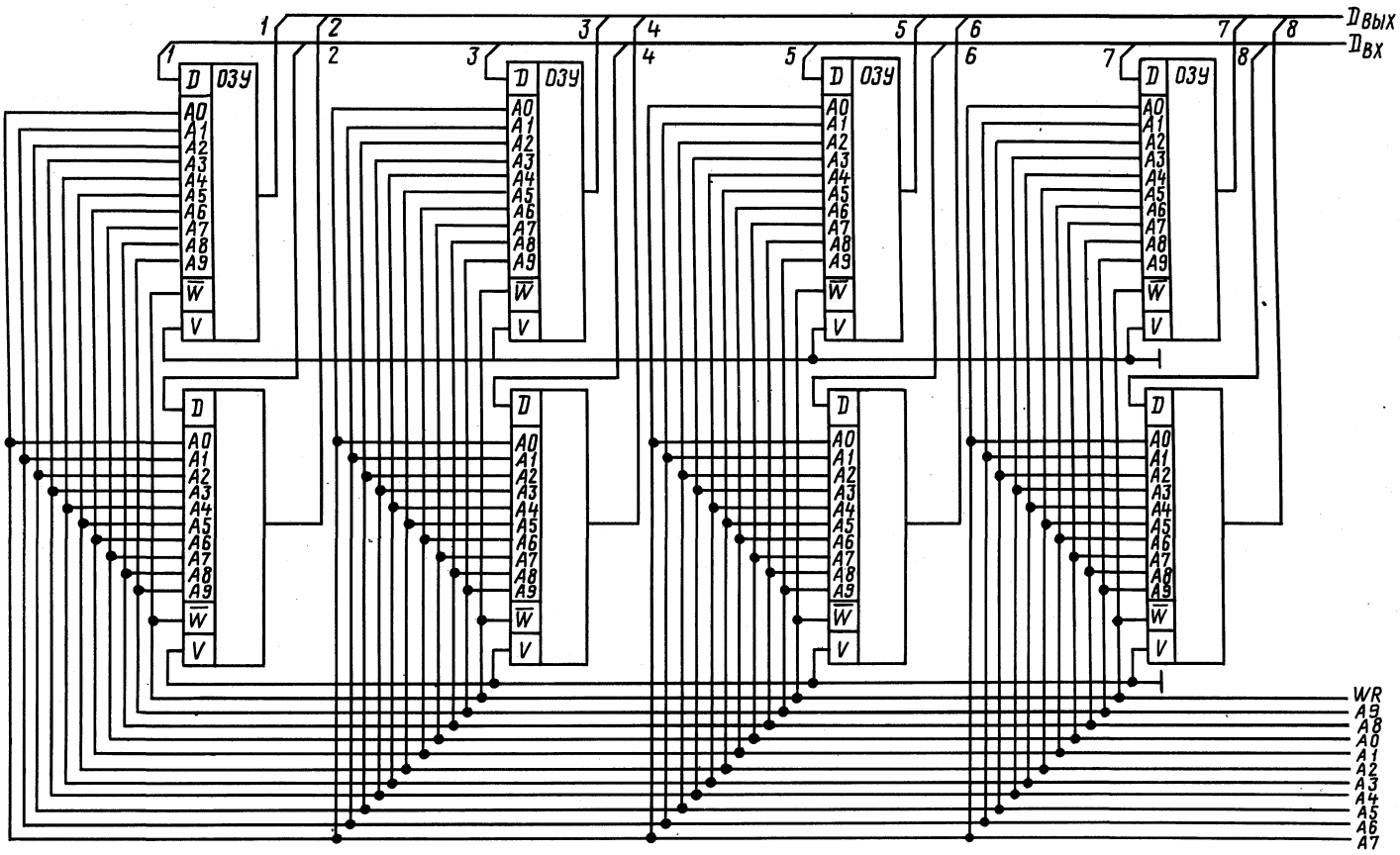


Рис.1. Структура ОЗУ микролаборатории

ПЗУ, программируемые масками, выполняются изготовителями интегральных схем. Такие ПЗУ часто используются в приборах, выпускаемых в больших объемах. Это объясняется тем, что они более дешевы и имеют больший объем памяти на одном кристалле.

Вторым типом является ППЗУ. Пользователь электрически программирует ПЗУ, используя специальное устройство, называемое программатором ППЗУ. Эти ППЗУ не могут перепрограммироваться.

Существуют также стираемые ППЗУ, подобные ПЗУ. Данные в них могут стираться и перепрограммироваться. Программируемые биты запоминаются как заряд конденсатора с нулевой утечкой.

Имеется также ПЗУ, где стирание выполняется ультрафиолетовым облучением через окошко в корпусе микросхемы. Эти устройства наиболее удобны для приборов, выпускаемых небольшими объемами.

Самым новым типом ППЗУ является электрически изменяемое ППЗУ, данные в котором могут электрически стираться в схеме. Достоинством такого типа ЗУ по сравнению со стираемым ППЗУ является то, что могут стираться небольшие секции. Электрически изменяемое ПЗУ не так легко использовать, как стираемое ПЗУ, кроме того они являются более дорогостоящими. Множество систем, требующих запоминания данных на длительный период времени, используют электрически изменяемые ПЗУ, но они должны время от времени восстанавливать эти данные. Электрически изменяемые ППЗУ применяются в цифровых устройствах настройки телевизоров, калиброванных передатчиках, устройствах автоматического набора телефонных номеров.

ПЗУ микролаборатории реализовано на двух биполярных микросхемах ППЗУ КР556РТ5 с организацией 512x8 битов /рис.2/. Предусмотрена возможность увеличения емкости ПЗУ до 1,5К байта, для чего необходимо добавить 1 микросхему в адаптер для ППЗУ.

768 байтам ПЗУ /в которых записана программа монитора/ отводятся адреса от 0000 до 02FF. Информация в этой области памяти не мо-

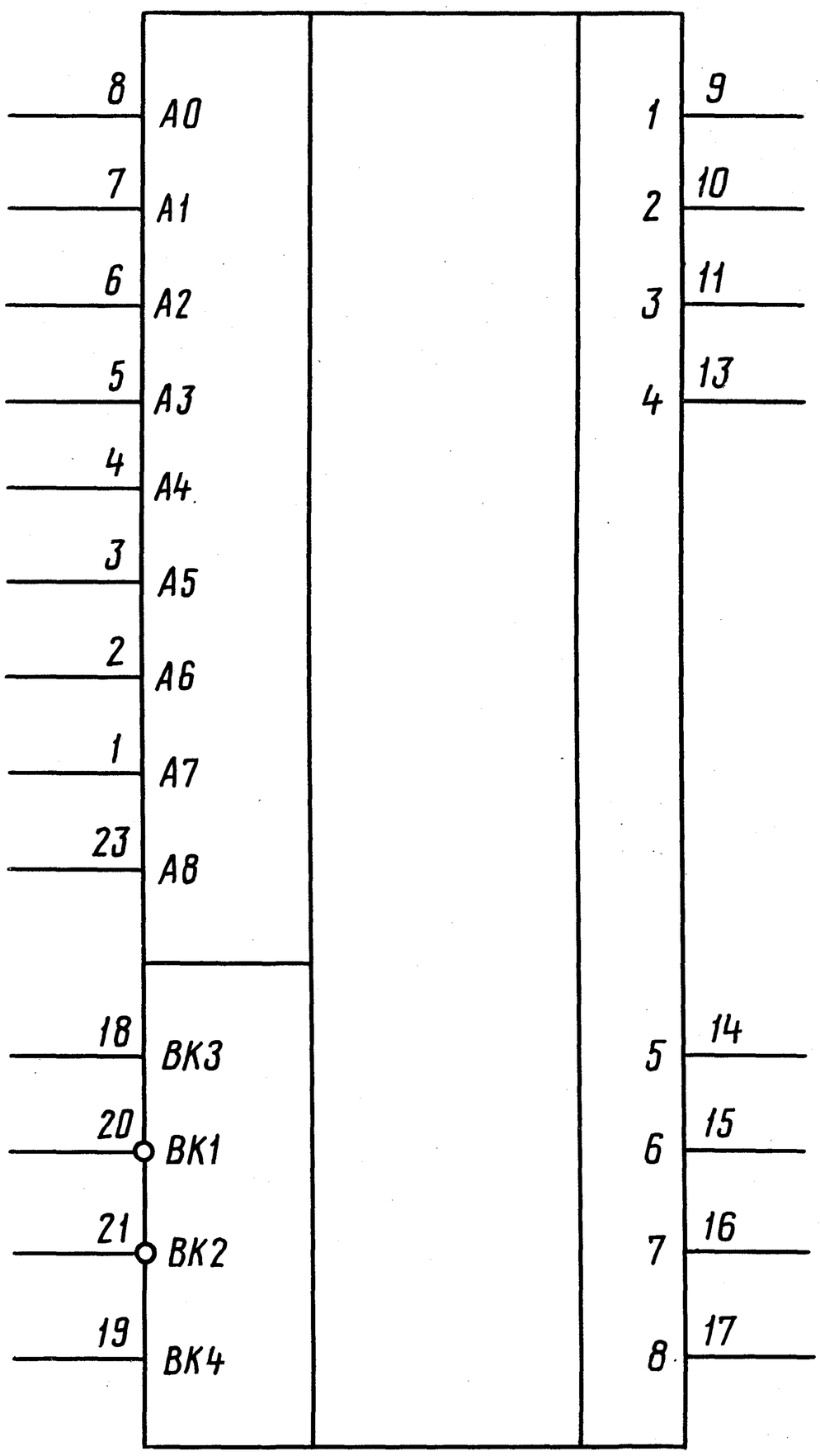


Рис.2. Схема ППЗУ микролаборатории

жет быть изменена пользователем с помощью перезаписи. Для изменения его содержимого необходимы новые микросхемы ППЗУ с занесенной в них постоянной программой пользователя, которая в этом случае будет играть роль монитора системы. Если же пользователю нужно расширить функции системы без изменения уже заложенных в нее возможностей, то его дополнительная постоянная программа может располагаться по адресам от 0300 до 03FF.

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА МИКРО-ЭВМ

Для того, чтобы микропроцессорная система могла быть полезной, она должна взаимодействовать с внешним миром, т.е. должна иметь, по крайней мере, одно входное и одно выходное устройство, подсоединяющиеся к системе через шину ввода/вывода. Устройства для ввода/вывода информации называются периферийными.

ПРОГРАММИРУЕМЫЙ ПЕРИФЕРИЙНЫЙ ИНТЕРФЕЙС И КЛАВИАТУРА

В "Микролаб" клавиатура сопряжена с системой через ППИ /БИС КР580ИК55/.

Средством управления всеми операциями системы является программа монитора. Все функции монитора задаются с помощью простых операций клавиатуры.

Микро-ЭВМ имеет на передней панели 25 клавиш и переключатели режима работы, которые обеспечивают прямую и удобную связь с микропроцессорной системой без каких-либо дополнительных аппаратных средств. Клавиатура имеет матричную организацию, кроме кнопки СБРОС. Опрос и обработка ее кодов производится программой монитора с помощью микросхемы D37 - ППИ. Микросхема D37 содержит три программируемых 8-разрядных шины ввода/вывода, соответствующие трем портам, и может работать в различных задаваемых программой режимах. В "Микролаб" ППИ устанавливается программой монитора в режим для сканирования клавиатуры.

Функционально этот режим обеспечивает следующую конфигурацию: две 8-разрядные шины /порт А и В/ и две 4-разрядные шины /порт С/. Каждая из них может работать как в режиме ввода, так и в режиме вывода, что определяется управляющим словом. В микро-ЭВМ 4 старших бита шины порта С программируются на вывод, а порт А- на ввод. Программа монитора выдает по очереди на С-выходы микросхемы D37 /выводы 13, 12, 11/ отрицательные импульсы, считывая после каждого вывода информацию с порта А /рис.3/. Каждый разряд порта С с С-выходов поступает через буферный каскад D8 на 8 кнопок /ряд/. Каждый вход порта А является сборкой трех кнопок из разных рядов /микросхемы D56-D58/. В случае, если ни одна из кнопок ряда, на который подан низкий уровень, не нажата, в порт А поступит код FF /шестнадцатеричный формат/, если же нажата хотя бы одна кнопка, то в соответствующем разряде

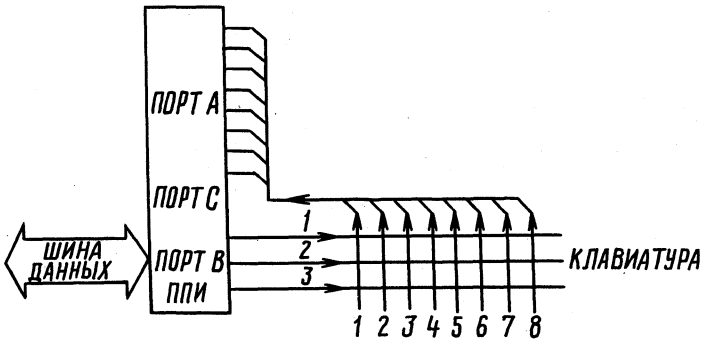


Рис.3. Клавиатурный интерфейс

порта А появится лог.0. Учитывая номер ряда и разряд порта А, в котором при вводе был лог.0, монитор формирует код нажатой кнопки и обращается в соответствии с ним к нужной подпрограмме, выполняющей соответствующую этой кнопке операцию.

К восьми разрядам порта В подключены светодиодные индикаторы. Этот порт будет программироваться как порт вывода.

К трем разрядам порта С /которые не используются при сканировании клавиатуры/ подсоединены тумблеры, т.е., запрограммировав ППИ определенным образом, можно считывать данные с тумблеров и выводить данные на светодиодные индикаторы.

К нулевому разряду порта В кроме светодиода подключен громкоговоритель, т.е., организовав определенным образом программу, можно "заставить" громкоговоритель звучать.

УСТРОЙСТВО ИНДИКАЦИИ

Выходным устройством, служащим для визуального наблюдения за внутренним состоянием системы, в микро-ЭВМ являются восемь восьмисегментных индикаторов табло VD1 - VD8. На них индицируются входные данные, адреса памяти, данные памяти, содержимое регистров центрального процессора ЦП и флагов состояний в соответствии с операциями клавиатуры. Информация отображается на индикаторах в шестнадцатеричном коде. Индикация информации динамическая. Данные на индикаторы передаются из восьми старших ячеек ОЗУ с адресами 83F8-83FF при помощи прямого доступа к памяти без участия ЦП. В каждой из этих ячеек ОЗУ находится восьмисегментный код, соответствующий своему индикатору.

Схема управления запросом шин состоит из генератора импульсов, собранного на транзисторах Т2, Т3, времязадающими элементами которого являются резисторы R56, R58 и конденсатор С6. На выходе микросхемы D54.4 /вывод 16/ формируется сигнал "Зхв.ш" /захват шины/, поступающий на ЦП /D11, вывод 13/, длительностью 3-5 мкс с периодом повторения 1-2 мс. Этот импульс также поступает на счетчик D51, используемый для сканирования адресов восьми ячеек ОЗУ, где запоминаются данные для индикатора и синхронного подключения питания к соответствующему индикатору. Пока ЦП находится в режиме захвата шин, вызванном этим импульсом, на его выходе /Птр.Зхв." /подтверждение захвата/ присутствует уровень лог.1. В это время данные для соответствующего индикатора поступают из соответствующей ячейки ОЗУ в триггер - защелку данных /D50/ по спаду сигнала "Зпр.Ш" /запрос шины/ и высвечиваются индикатором. Подобным образом с интервалом 1-2 мс высвечиваются все 8 индикаторов.

ПЕРИФЕРИЙНЫЕ ИНТЕРФЕЙСНЫЕ КРИСТАЛЛЫ

С использованием разнообразных БИС /кристаллов ввода/вывода/ упрощается сопряжение, уменьшается количество компонентов, уменьшается стоимость и увеличивается эффективность системы. Самым простым из таких кристаллов является рассмотренный выше ППИ. Он представляет собой 40-выводную ИС, содержащую 3 шины ввода/вывода /рис.4/. Каждая из шин используется как входная или выходная шина. Направление каждой шины управляется регистром на кристалле. Программа инициализации, находящаяся в ПЗУ системы, устанавливает управляющий регистр в нужную комбинацию входных и выходных шин. Такие БИС являются очень гибкими, так как каждая шина может изменяться программой. Кроме того, они обычно включают управляющую логику для синхронизации работы.

Другой общий тип интерфейсных кристаллов, обеспечивающих после-

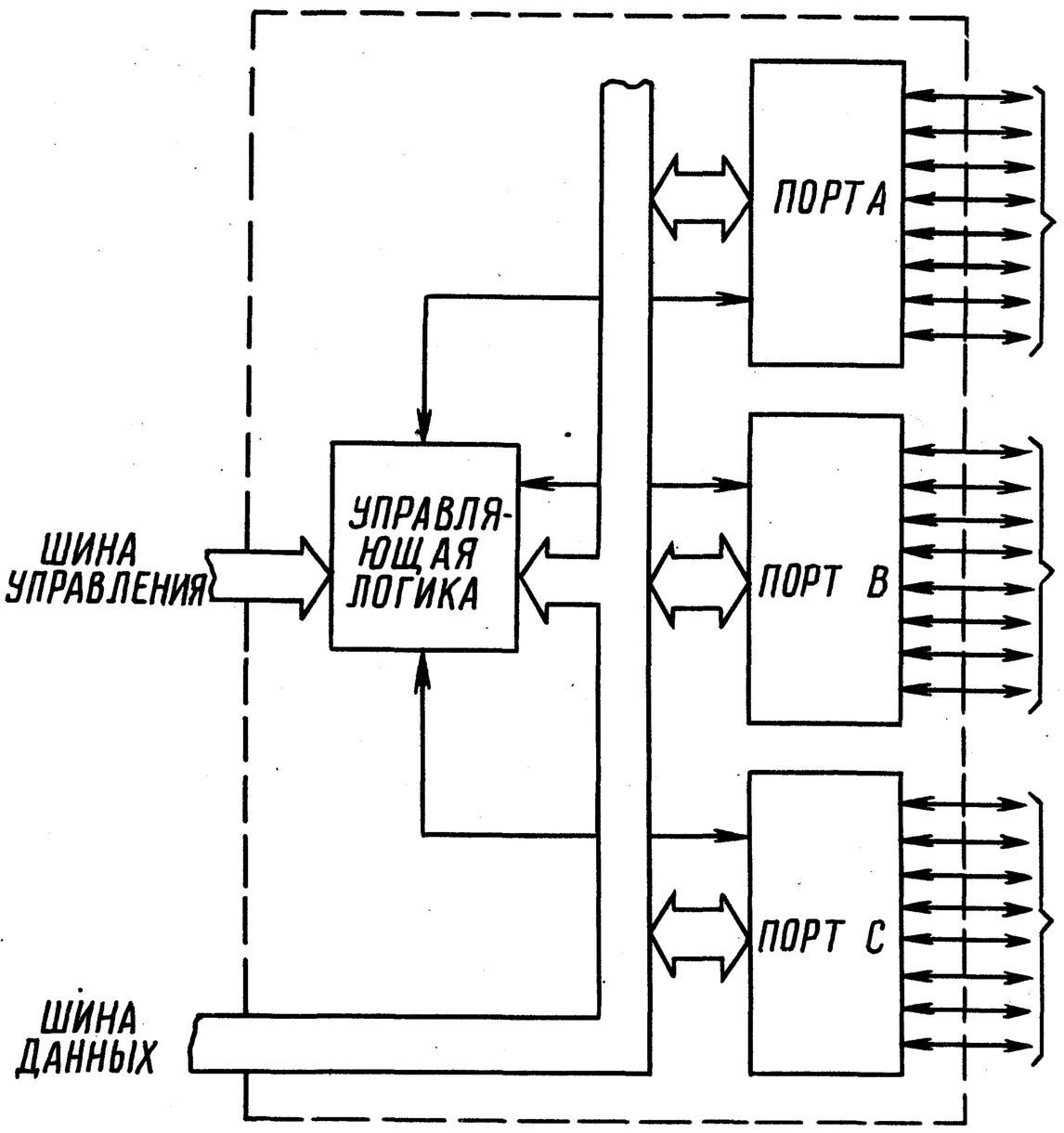


Рис.4. Периферийное интерфейсное устройство, содержащее три 8-разрядных шины ввода/вывода

довательные входы и выходы /рис.5/, универсальные асинхронные передатчики и приемники. Они принимают байт данных с МП, а затем выдают его по одному биту, т.е. действуют подобно регистру параллельно-последовательного ввода/вывода. Кроме того могут автоматически вводиться биты старта, останова и другие синхронизирующие и управляющие сигналы.

Формат управляется управляющим регистром, подобным рассмотренному для ППИ. Рассматриваемые интерфейсы

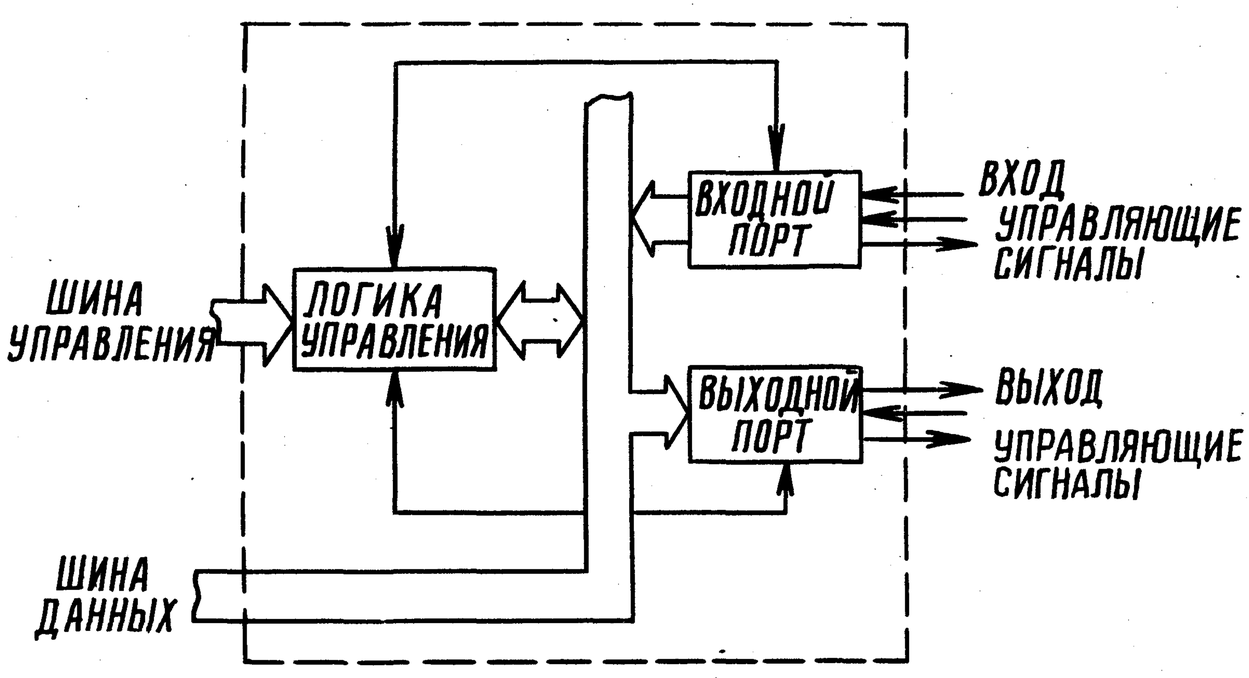


Рис.5. Интерфейс, обеспечивающий последовательную связь между двумя системами

также могут обрабатывать данные в другом направлении, преобразовывая поток последовательных битов в параллельную форму, подходящую для непосредственного использования МП. Обычно для связи между микропроцессорной системой и периферийным устройством, таким как ЭЛТ или телетайп, используется последовательный ввод/вывод. Благодаря тому, что информация имеет последовательный формат, для соединения устройств необходимы только два провода.

Имеются также разнообразные специализированные интерфейсные кристаллы, включающие контроллеры гибких дисков, контроллеры ЭЛТ, контроллеры ЗУ с прямым доступом, контроллеры клавиатур и дисплея. Многие из этих БИС являются даже более сложными, чем МП. Некоторые из них используют для управления внутренними функциями МП общего назначения.

Урок 10

СИГНАЛЫ УПРАВЛЕНИЯ В МИКРОПРОЦЕССОРНОЙ СИСТЕМЕ

СХЕМЫ УПРАВЛЕНИЯ

Схемы управления являются основными функциональными блоками центрального процессора. Используя входы синхронизации, схемы управления поддерживают надлежащую последовательность событий, необходимых для любой задачи обработки. После того, как команда выбрана и декодирована, схемы управления выдают соответствующие сигналы /в устройства, которые по отношению к центральному процессору являются как внутренними, так и внешними/, для того, чтобы начать обработку. Часто схемы управления должны отвечать на внешние сигналы, например, запросы на прерывания.

Для выполнения команд МП КР580ИК80 требуется 1-5 машинных циклов. В начале каждого цикла на шину данных выдается байт состояния /в течение действия сигнала СИНХР./ Так как информация о состоянии выдается из ЦП на шину данных в тече-

ние коротких интервалов времени, то необходимо позаботиться о ее хранении. В типичной структуре системы КР580ИК80 для хранения байта состояния и его дешифрации используется специальный системный контроллер.

В микролаборатории системный контроллер состоит из двунаправленного шинного формирователя, подключенного к системной шине данных, и регистра состояний D14, выходы которого соответствуют состоянию ЦП в текущем машинном цикле. Стробирование регистра производится в момент прихода синхроимпульса Ф1 /ТТЛ/ и синхросигнала СИНХР., т.е. во время выдачи процессором на шину данных информации о состоянии. Из этой информации и сигналов ЦП /ВЫДАЧА, "Птр.зхв", ПРИЕМ/ получают стандартные команды микропроцессорной системы: "Птр.пр" /подтверждение прерывания/, "Чт." /чтение/, "Зп." /запись/, "Чт.ЗУ", "Зп.ЗУ".

ДРУГИЕ УПРАВЛЯЮЩИЕ СИГНАЛЫ

Синхроимпульсы

Действия центрального процессора являются циклическими. Процессор выбирает команду, выполняет требуемые операции, выбирает следующую команду и т.д. Регулярная последовательность событий, подобная указанной, требует синхронизации, поэтому центральному процессору требуется тактовый генератор, который синхронизирует работу процессора.

В микролаборатории генератор собран на трех инверторах D1. Времязадающим элементом является емкость С, которая задает частоту генерирования, равную 18 МГц. С выхода генератора сигнал поступает на вход С сдвигового регистра D3, выполняющего роль делителя входной частоты. Коэффициент деления равен 9 и определяется обратными связями с выходов регистра через микросхему /8И-НЕ/ на вход Р. На каждом выходе регистра появляются отрицательные импульсы, сдвинутые относительно соседнего разряда на t=1/18 мкс и с частотой следования 2 МГц /18/9 МГц/. Выходные импульсы с 1-го и 0-го разрядов регистра поступают на вентиль и формируют синхроимпульс Ф1 /ТТЛ-уровня/. Синхроимпульс Ф2 /ТТЛ/ формируется сигналами, поступающими с 3-го и 7-го разрядов регистра на RS-триггер.

Сигнал УСТАНОВКА

В блок центрального процессора входит схема формирования сигнала УСТАНОВКА. Сброс может выполняться двумя способами: автоматический сброс при включении питания и ручной сброс, осуществляемый нажатием кнопки СБРОС на передней панели, при этом сигнал лог.0 поступает на D-вход триггера D4 и проходит /с приходом на Т-вход тактового импульса f=2 МГц/ на выход, где держится в течение времени, кратного периоду синхроимпульса Ф2.

Сигнал ГОТОВНОСТЬ

Как было отмечено, действия процессора синхронизируются основным тактовым генератором.

Период синхронизации регулирует распределение времени на все действия обработки. Однако скорость цикла обработки ограничена временем обращения к памяти. Если процессор послал адрес считывания в запоминающее устройство, то он не может продолжать работу, пока не ответит запоминающее устройство. Многие запоминающие устройства могут отвечать гораздо быстрее, чем требует цикл обработки. Однако некоторые не могут выдавать адресуемый байт в минимальное время, устанавливаемое тактовым генератором процессора. Поэтому процессор должен иметь средства для синхронизации, которые позволяют запоминающему устройству запрашивать состояние ожидания. Если запоминающее устройство принимает сигналы СЧИТЫВАНИЕ или ЗАПИСЬ, то оно устанавливает сигнал запроса в линии ГОТОВНОСТЬ процессора, временно прерывая его работу. После того, как запоминающее устройство получило время для ответа, оно освобождает линию ГОТОВНОСТЬ процессора, и цикл команды продолжается.

Когда вход сигнала ГОТОВНОСТЬ переходит в низкое состояние, МП переходит в состояние ожидания. В этом состоянии шины остаются до

тех пор, пока сигнал ГОТОВНОСТЬ снова не перейдет в высокое состояние. Состояние ожидания позволяет применять медленные ЗУ и устройства ввода/вывода.

В микролаборатории выход сигнала подтверждения того, что ЦП находится в состоянии ожидания, соединен с входом сигнала ГОТОВНОСТЬ. Таким образом, МП находится в состоянии ожидания в течение одного такта, т.е. в течение одного периода синхросигнала.

Сигнал ЗАХВАТ ШИН

Важным свойством, улучшающим производительность процессора, является способность выполнять пересылки с прямым обращением к памяти. В обычных операциях ввода и вывода процессор сам управляет всей передачей данных. Информация, которая должна быть размещена в памяти, пересылается из устройства ввода в процессор, а затем из процессора в указанную ячейку памяти. Аналогично через процессор проходит информация, которая передается из памяти в устройства вывода.

Однако некоторые периферийные устройства способны передавать информацию в память и из памяти настолько быстро, что сам процессор может замедлять пересылку. Если в такое устройство или из устройства должно пересылаться какое-нибудь значительное количество данных, то производительность системы может быть увеличена имеющимися устройствами, которые выполняют прямую передачу данных. Во время такой передачи процессор должен временно приостановить работу, чтобы предотвратить конфликты, которые могут возникнуть, если процессор и периферийное устройство попытаются одновременно использовать канал системы.

Прямой доступ к памяти - это метод переноса данных непосредственно между памятью и периферийным устройством. Данные через МП не проходят. Это обеспечивается управляющим входом "Зхв.ш" МП. Когда эта линия находится в высоком состоянии, МП заканчивает машинный цикл, выполняемый в данное время, и затем останавливается, чтобы перевести линию "Птр.зхв." в высокий уровень. Все выходы шин МП /адресной, данных и управления/ переводятся в высокоимпедансное состояние. Теперь периферийные устройства могут управлять этими шинами и выполнять все нужные переносы данных. Когда линия "Зхв.ш." устанавливается периферийным устройством в низкое состояние, МП продолжает работу с той точки, где он остановился.

Для выдачи адресных и управляющих сигналов периферийное устройство должно иметь управляющее устройство для прямого доступа к памяти. Имеются кристаллы контроллера прямого доступа к памяти /ПДП/, выполняющие эту функцию.

Если МП системы имеет ПДП, то все буферы шин должны иметь три состояния. Линия "Птр.зхв" запрещает выходы буферов и позволяет контроллеру ПДП использовать эти шины.

ПРЕРЫВАНИЯ

В качестве способа улучшения работы процессора во многие ЦП включаются средства прерывания. Прерывания позволяют внешней аппаратной части запросить у процессора немедленное выполнение каких-то действий. Они прерывают обычный ход программы и передают управление на специальную программу матобеспечения.

Рассмотрим работу вычислительной машины, которая обрабатывает большие объемы данных, часть из которых должна быть выведена на печать. За один машинный цикл ЦП может вывести один байт данных, но при фактической печати знака, определяемого байтом данных, необходимое время может оказаться равным многим машинным циклам. В результате ЦП будет оставаться свободным, ожидая,пока печатающее устройство сможет принять следующий байт данных. Если в вычислительной машине осуществлена возможность прерывания, то ЦП, выведя байт данных, может затем возвратиться к обработке данных, а когда печатающее устройство готово к приему следующего байта данных, то оно может выдать запрос на прерывание. Когда ЦП опознает прерывание, то он приостановит выполнение основной программы

и автоматически перейдет к программе, которая будет выводить следующий байт данных. После вывода байта данных ЦП продолжает выполнение основной программы. Заметим, что в принципе это совершенно аналогично вызову подпрограммы, за исключением того, что переход индицируется извне, а не программой.

Возможны более сложные структуры прерываний, в которых несколько устройств, требующих прерывания, используют один и тот же процессор, но имеют различные уровни приоритетов. Обработка с прерываниями является важным свойством, которое дает возможность максимально использовать мощность процессора.

В ЦП КР580ИК80 имеется триггер разрешения прерывания, который может быть установлен или сброшен. Если триггер находится в нулевом состоянии, то система прерываний блокирована и запросы прерываний не воспринимаются. Когда ЦП воспринимает запрос прерывания от внешнего устройства, происходит следующее:

1 - заканчивается выполняемая команда;

2 - сбрасывается бит разрешения прерывания;

3 - прерывающее устройство формирует команду и посылает ее в ЦП для выполнения. Команда не размещена в памяти, и программист не может ее изменить, это является функцией разработчика системы прерываний.

Прерывающее устройство обычно формирует команду RST, представляющую собой команду вызова одной из восьми 8-байтовых подпрограмм, расположенных в первых 64 ячейках памяти. Вход запроса прерывания осуществляет это прерывание. Схема, запрашивающая прерывание, выдает код на шину данных, указывая адрес обслуживания прерывания, на который и совершается переход. В микролаборатории прерывание обеспечивает работу системы в шаговом режиме. Тумблер устанавливается в положение ШАГ, и при нажатии кнопки ПУСК сигнал приходит на вход запроса прерываний МП. МП, выполнив одну команду, останавливает дальнейшее выполнение программы, которое не возобновляется до тех пор, пока не будет нажата кнопка ВОЗВРАТ. В это время выполняется подпрограмма обработки прерывания и происходит вывод на дисплей содержимого аккумулятора и регистра состояния /флагов/.

НАГРУЗКА

Одним из факторов, гарантирующих работу логики является выходная нагрузка. Выход вентиля может подавать только ограниченный по величине ток и, если схема вынуждена выдавать больший ток, то она перестает правильно действовать. Если схема значительно перегружена, то логические уровни могут измениться, а вентиль перегревается и сгорает.

В микропроцессорных системах обычно рассматривают два типа нагрузки: статическую и динамическую. Статическая нагрузка обусловливается резистивными и токовыми компонентами логических узлов. Динамическая нагрузка, в основном, является результатом емкости узла. Статическая нагрузка выходов МОП, запускающих множественные ТТЛ-входы, рассматривается в первую очередь. Тем не менее, когда на шине находится большое количество устройств, и шина проходит через разъемы к множеству плат, может стать важной динамическая нагрузка, влияющая на быстродействие и синхронизацию.

В микропроцессорной системе к одной шине подсоединяется множество устройств, поэтому важно учитывать не только динамическую нагрузку, но и статическую, которая влияет на уровни логического напряжения и границы шума. Адресные выходы МП могут запускать все устройства, подсоединенные к адресной шине. Каждое устройство, запускающее шину данных, должно запускать все устройства, подсоединенные к шине.

Микролаборатория использует буферы на всех адресных линиях и всех линиях шины данных, потому что МП не имеет достаточного по величине выходного запускающего тока. Для согласования с адресной шиной в микролаборатории используется однонаправленный буфер, так как адреса проходят по адресной шине в одном направлении. Поскольку по шине данных данные проходят в обоих направлениях, шина данных согласуется двунаправленным буфером.

ШУНТИРОВАНИЕ

Во всех цифровых системах нужно следить, чтобы цепи питания и заземления пропускали как можно меньше помех. Шунтирующие конденсаторы размещены на всей печатной плате. Кроме того, для 5-вольтовых источников на печатную плату помещается танталовый конденсатор емкостью не менее 10 мкФ.

СИНХРОНИЗАТОР

Чтобы микропроцессорная система работала правильно, все режимы работы должны быть синхронизированы. Большинство из них управляется встроенным МП. Поэтому поток сигналов между устройствами в системе должен координироваться. Все сигналы, поступающие на адресные шины, шины данных и управления, строго синхронизированы.

На рис.1 показана синхронизация операции записи. Перед выполнением любой операции адрес должен быть стабилен некоторое время /время выборки/. Это позволяет дешифраторам внутреннего адреса памяти выбирать указанные ячейки памяти. Данные должны быть стабильными определенный период времени /называемый временем установки/ перед выполнением записи, а также после импульса записи /время задержки/.

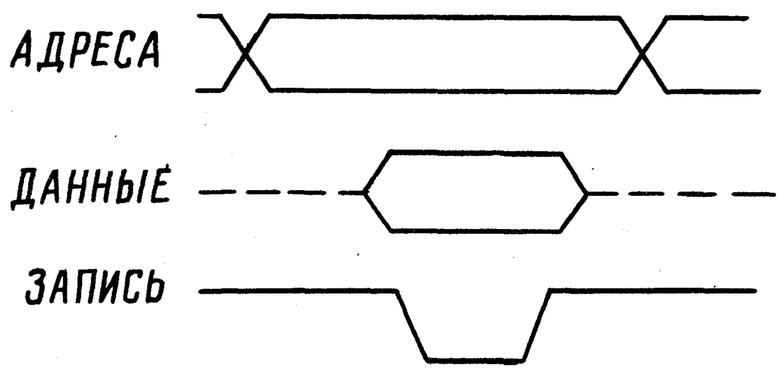


Рис.1. Данные, занесенные в память фронтом сигнала ЗАПИСЬ

На рис.2 показана синхронизация операции считывания. Как и при операции записи, адрес должен быть стабилен достаточно долго для того, чтобы внутренний дешифратор в ЗУ установился. Затем генерируется сигнал считывания, и после определенного периода /времени выборки данных/ память переводит адресуемые данные на шину данных. Эти данные должны быть стабильными на время установки перед фронтом сигнала

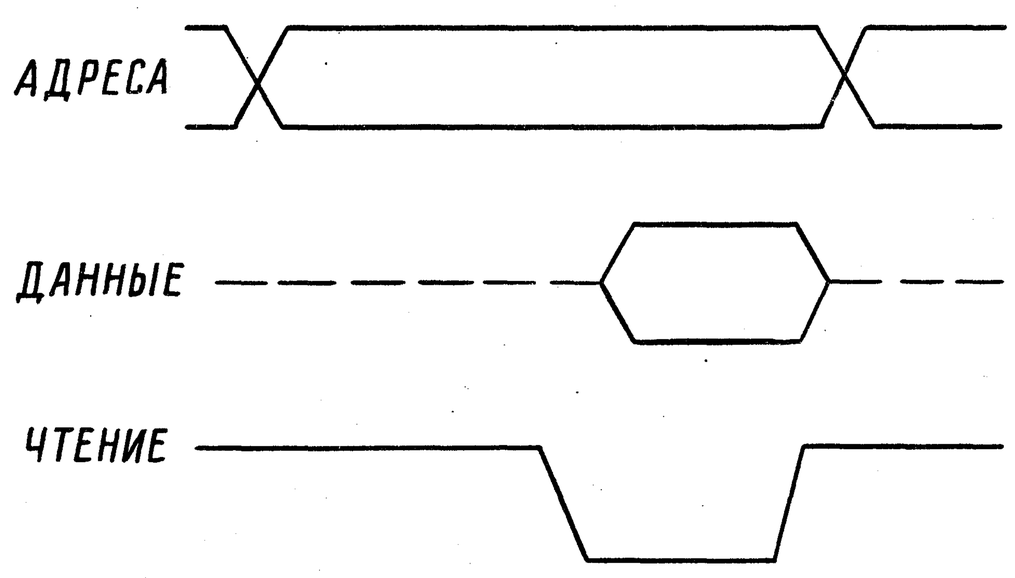


Рис.2. Считывание данных из МП на фронте сигнала ЧТЕНИЕ

ЧТЕНИЕ, когда данные считываются МП. Данные должны также оставаться стабильными на период задержки данных.

На рис.3. показана синхронизация центрального процессора для типичной команды. Основной единицей синхронизации является состояние, которое длится один период синхроимпульса. Один машинный цикл состоит из 3-5 состояний. Большинство простых операций требует одного машинного цикла. Цикл команды - это время, необходимое для выполнения всей команды. Он состоит из 1-5 машинных циклов.

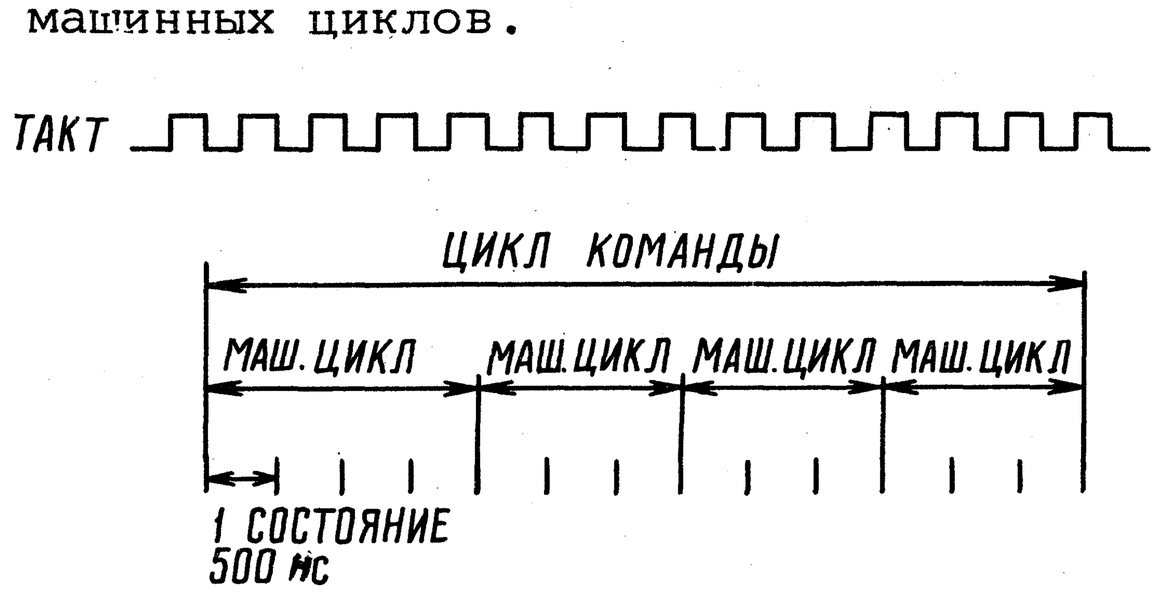


Рис.3. Типичный цикл команды

ПРОГРАММНЫЕ СРЕДСТВА

Урок 11

РЕГИСТРЫ И ТОЧКИ ПРЕРЫВАНИЯ

ОПИСАНИЕ КОМАНД

Для описания команд введены некоторые сокращения. Термин „данные введен для обозначения восьми разрядов данных, „адр". - для обозначе-

ния 16 разрядов адресов /полное описание команд МП КР580ИК80 приведено в приложении/.

Команды манипулирования с данными MVI, INR, CMA

Одной из основных задач МП является загрузка аккумулятора данными. Это выполняется командой MVI A [данные] /непосредственная загрузка аккумулятора/. Данные, загружаемые в аккумулятор, запоминаются в байте, следующим за кодом.

После того,как данные введены в аккумулятор, необходимы команды для работы с ними. Ранее использовались две команды: INR A /приращение содержимого аккумулятора/, CMA /инвертирование содержимого аккумулятора/.

Команды проверки и перехода CPI JZ, JMP

Чтобы проверить содержимое аккумулятора, используется команда CPI [данные] /непосредственное сравнение/. При этом сравниваются данные, указанные во втором байте команды, с содержимым аккумулятора,и, соответственно, устанавливаются флаги процессора. До сих пор использовался только нулевой флаг, который устанавливается, если в результате операции получается нуль. Команда JZ [адр.] /переход, если 0/ проверяет нулевой флаг, предварительно установленный предыдущей командой, такой как CPI, и вызывает переход при установленном флаге. Команда безусловного перехода JMP [адр.] вызывает переход независимо от состояния флага. Адрес перехода обеих команд запоминается в двух байтах памяти, следующих за кодом операции.

Команды передачи между аккумулятором и памятью LDA, STA

Команды LDA [адр.] /загрузка аккумулятора/ и STA [адр.] /запоминание содержимого аккумулятора/ передают данные между аккумулятором и памятью. Адрес ячейки памяти указывается в двух байтах, следующих за операционным кодом.

Команды связи с подпрограммой CALL, RET

Для использования подпрограмм требуется две команды. CALL [адр.] используется для перехода к подпрограмме, а RET /возврат/ используется для окончания подпрограммы. Команда CALL указывает адрес так же, как команда перехода. Команда RET не указывает адреса, но вызывает переход к команде, следующей за предварительно выполненной командой CALL.

Команды управления прерываниями EI, DI

Команда EI /разрешение прерываний/ разрешает выбранные прерывания. Команда D1 /запрещение прерываний/ запрещает все прерывания.

Относительно небольшой набор команд, перечисленных выше, характеризует основные возможности МП КР580ИК80. Однако система команд этого МП включает значительно большее число команд, что облегчает написание программ. Аналогично при конструировании аппаратной части можно осуществить любую логическую цепь при использовании вентилей И-НЕ. Однако аппаратная часть упрощается с использованием таких устройств, как вентили ИЛИ-НЕ, триггеры, мультиплексеры, счетчики и сумматоры.

РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ

В МП КР580ИК80 имеются шесть 8-разрядных регистров, которые используются для временного запоминания данных. На рис.1 показана структурная схема МП КР580ИК80, включающая эти регистры /B,C,D,E,H и L/ и регистр - указатель стека.

Для использования этих регистров необходимы некоторые новые команды. Команда MVI, загружающая данные в аккумулятор, может быть использована с любым регистром. Например, MVI D [данные] обеспечивает запись данных в регистр D. Общей формой записи этой команды является MVI r [данные], где r - любой регистр /A, B, C, D, E, H, L/. Хотя аккумулятор является специальным в смысле использования для

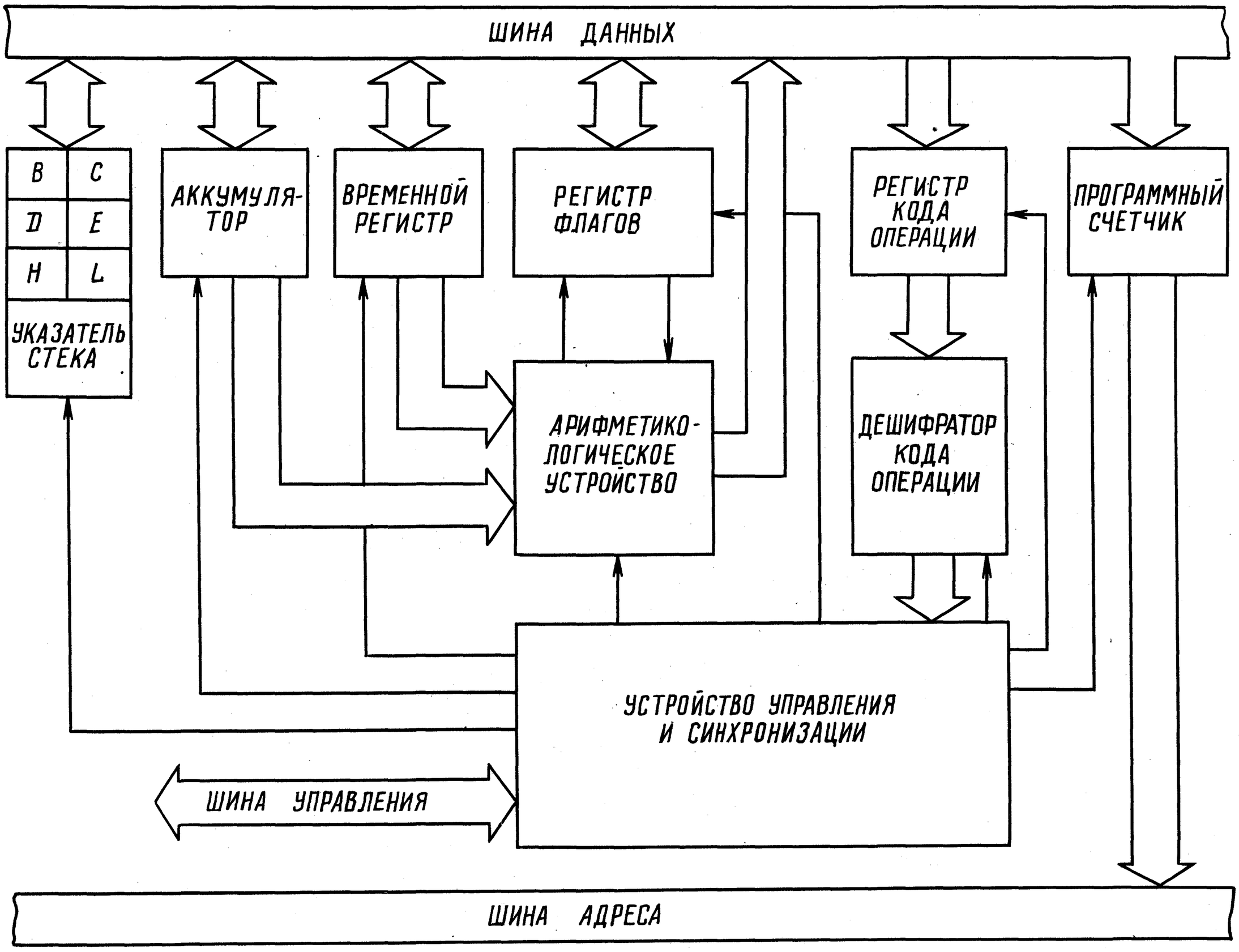


Рис.1. Структурная схема МП КР580ИК80

результатов вычислений, он также может быть использован как регистр общего назначения.

Команда INR также может быть использована с любым регистром. Общей формой записи является INR r. Например, INR H наращивает содержимое регистра H.

Данные можно перемещать из одного регистра в другой. Это выполняется командой MOV r1, r2.Регистр r1 является местом назначения, а r2 - источником. Например, MOV A,H переносит содержимое регистра H в аккумулятор, но содержимое регистра H не меняется.

Регистры общего назначения удобны, когда программа использует несколько разных переменных. Каждый регистр может быть использован для определенной цели. Нет необходимости использовать ячейки ОЗУ для запоминания данных до тех пор, пока достаточно 6 регистров. Например, программа, которая считает 6 разных событий, может использовать один регистр для счета каждого события.

КОСВЕННАЯ АДРЕСАЦИЯ

В приложении в командах MOV записан регистр M. Однако это не реальный регистр. Так обозначается ячейка памяти, адрес которой запоминается в регистрах H и L. Такая адресация называется косвенной, т.е. команда указывает, где запоминается адрес, а не действительный адрес /в данном случае регистры H и L/. Например, если регистр H содержит 12, а регистр L - 37 /рис.2/ команда MOV A, M будет загружать в аккумулятор содержимое ячейки памяти с адресом 1237. Результат будет тот же, как при команде LDA 1237. Это является примером того, как одна операция может быть выполнена двумя способами. MOV A, M явля-

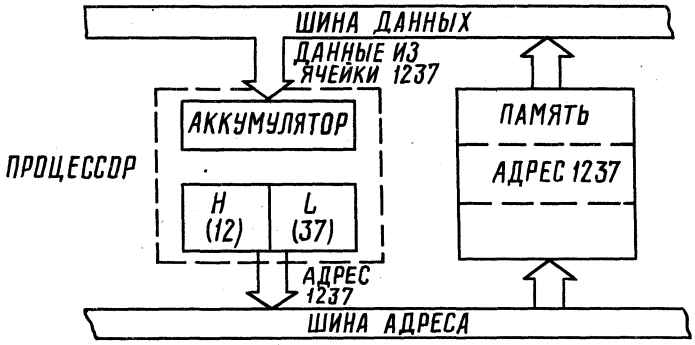


Рис.2. Косвенная адресация с использованием H и L регистров

ется однобайтовой командой, но требует предварительной загрузки регистров H и L. С другой стороны, LDA 1237 является 3-байтной командой. Однако ее часто предпочитают, так как она не требует запоминания адреса в регистрах H и L. Косвенная адресация особенно удобна при работе с таблицами.

Проведем эксперимент для демонстрации использования регистров общего назначения при работе команды MOV.

1. Введите программу, приведенную в табл.1 /программа сначала заносит в регистр B число 37, затем переносит его содержимое в регистр H, наконец, наращивает регистр H/.

Таблица 1

Примеры программы

Адреса	Данные	Команды	Комментарии
8000	06	MVI B, 37	Загрузка числа 37 в регистр B
8001	37		
8002	60	MOV H,B	Загрузка регистра H содержимым регистра B
8003	24	INR H	Приращение содержимого регистра H

2. Удостоверьтесь, что программа правильно записана в память.

3. Установите тумблер АВТ-ШАГ в положение ШАГ.

4. Нажмите последовательно кнопки 8, 0, 0, 0, УСТ.АД.

5. Нажмите кнопку ПУСК. При этом выполняется команда MVI B,37. На индикаторе адресов высвечивается число 8002 - это адрес следующей команды, записанный в программном счетчике. На индикаторе данных в двух правых разрядах высвечивается содержимое флагового регистра, а в в двух левых - содержимое аккумулятора.

6. Нажмите последовательно кнопки 8, 3, E, 9, УСТ.АД. На индикаторе адресов установится адрес регистра B. На индикаторе данных в двух правых разрядах высвечивается число 37 - содержимое регистра B после выполнения команды MVI B,37.

Таблица 2

Программа счетчика

Адреса	Данные	Команды	Комментарии
8000	21	LXI H, 83F7	Загрузка в регистр /L, H/ адреса ячейки высвечиваемого индикатора
8001	F7		
8002	83		
8003	36	MVI M,O	Очистка ячейки высвечиваемого индикатора
8004	00		
8005	CD	CALL SEG	Вызов подпрограммы высчечивания символа
8006	CO		
8007	01		
8008	21	LXI H, 83F7	Восстановление содержимого регистра /L, H/
8009	F7		
800A	83		
800B	D7	RST 2	Точка прерывания
800C	34	INR M	Приращение содержимого ячейки высвечиваемого индикатора
800D	C3	JMP 8005	Переход к программе высвечивания символа
800E	05		
800F	80		

Подпрограмма обработки прерывания

8015	CD	CALL KEY IN	Вызов подпрограммы опроса клавиатуры
8016	16		
8017	02		
8018	FE	CPI 1O	Проверка: нажата ли кнопка ПУСК
8019	10		
801A	CA	JZ 800C	Переход к приращению содержимого ячейки высве-
801B	OC		

Продолжение табл.2

Адреса	Данные	Команды	Комментарии
801C	80		чиваемого индикатора
801D	C3	JMP 8005	Переход к программе высвечивания прежнего символа
801E	05		
801F	80		
83D1	C3	JMP 8015	Переход к подпрограмме обработки прерывания
83D2	15		
83D3	80		

7. Нажмите кнопку ВОЗВР. На индикаторе адресов высвечивается адрес 8003. При этом выполнится команда MOV H, B.

8. Установите на индикаторе адресов число 83E5 - адрес регистра H. На индикаторе данных в двух правых разрядах высвечивается число 37 - содержимое регистра H после выполнения команды MOV H, B. Установив на индикаторе адресов адрес регистра B, убедитесь, что этот регистр все еще содержит число 37.

9. Нажмите кнопку ВОЗВР. На индикаторе адресов высвечивается число 8004 - адрес следующей команды. При этом выполняется команда INR H. Установив на индикаторе адресов адрес регистра H, убедитесь, что его содержимое увеличилось на 1, т.е. стало равно 38.

ТОЧКИ ПРЕРЫВАНИЯ

Пошаговое выполнение программы позволяет останавливать ее между командами и наблюдать действие каждой команды. Но во многих случаях удобно выполнять программу на полном быстродействии и останавливать в интересующей точке. Это можно осуществить вводом в программу такой команды, которая заставляет процессор прерывать исполнение команды пользователя и перейти на программу прерывания.

"Микролаб" использует команду RST для выполнения прерывания. Она подобна команде CALL, за исключением того, что не явно определяет адрес. RST 2 эквивалентна CALL 0010. Адрес 0010 фиксируется при конструировании МП КР580ИК80 и не может быть изменен. В "Микролаб" в программе монитора по этому адресу записан переход на определенный адрес в данном случае 83D1. По адресу 83D1 необходимо записать переход на подпрограмму обработки прерывания.

ИСПОЛЬЗОВАНИЕ ТОЧЕК ПРЕРЫВАНИЯ

Точки прерывания удобно использовать при отладке программ. Эти точки можно вводить после написания программы путем замены некоторой команды точкой прерывания. Точка прерывания останавливает программу в нужном месте, поэтому не надо выполнять предыдущие команды по шагам. Если программа выполняется правильно, точка прерывания может быть заменена NOP или командой, которую заменила точка прерывания. В "Микролаб" прерывание RST 1 выводит программу пользователя на монитор.

Аппаратные точки прерывания

Кроме программного прерывания в микро-ЭВМ может быть реализовано аппаратное прерывание.

В "Микролаб" такое прерывание используется для организации шагового режима. Если тумблер режима находится в положении ШАГ и нажимается кнопка ПУСК, то на МП поступает сигнал прерывания, который воспринимается им как RST 7, /это обеспечивается аппаратной частью/. Подпрограмма обработки прерывания RST 7 является подпрограммой шагового режима.

Проведем эксперимент по использованию точки прерывания.

1. Введите программу, приведенную в табл.2. Основная программа, записанная в ячейках 8000-800F. представляет собой программу счетчика с точкой прерывания. Счет ведется в ячейке 83F7. Содержимое этой ячейки выводится на крайний правый индикатор. Перед каждым приращением счетчика происходит прерывание программы, и начинает выполняться подпрограмма с адреса 8015. В этой подпрограмме предусмотрен опрос клавиатуры, и если нажата кнопка ПУСК, происходит возврат в основную программу, где

произойдет очередное приращение ячейки 83F7. Таким образом, после каждого нажатия кнопки ПУСК будет происходить увеличение на 1 содержимого ячейки 83F7.

2. Установите тумблер режима в положение АВТ и, нажимая кнопку ПУСК, наблюдайте на правом индикаторе счет данных.

Урок 12

НАБОР КОМАНД МИКРОПРОЦЕССОРА КР580ИК80

ЛОГИЧЕСКИЕ КОМАНДЫ

В основе построения цифровых схем лежит логический вентиль, а наиболее широко используются четыре базовые функции вентиля: "НЕ", "И", "ИЛИ", "Исключающее ИЛИ". Каждая из этих функций может быть реализована программным способом.

Функцию "НЕ" выполняет команда CMA /дополнение аккумулятора/. Каждый бит аккумулятора после исполнения этой команды инвертируется.

Функцию "И" выполняет команда ANA R. Символ R обозначает либо один из регистров, либо ячейку памяти. Например, ANA D вызывает объединение по "И" содержимого регистра D и аккумулятора. Результат остается в аккумуляторе, содержимое регистра /в нашем примере D/ не изменяется. Операция с каким-то определенным регистром имеет свой код.

Рис.1 показывает аппаратный эквивалент команды ANA R. Функция "И" выполняется индивидуально для каждого бита аккумулятора. Например, если A=1011 0110 и D=0011 1100, результатом команды ANA D будет:

```
   0011 1100 - D
И  1011 0110 - A
   ---------
   0011 0100 - A
```

Функцию "ИЛИ" выполняет команда ORA R /объединение по "ИЛИ" аккумулятора и регистра R/.

Функцию "Исключающее ИЛИ" выполняет команда XRA R. Выполнение этих команд подобно выполнению ANA R, только реализуются другие логические функции. Поскольку адрес или данные определять не нужно, то все эти команды требуют только один байт кода операции /см.набор команд МП КР580ИК80/.

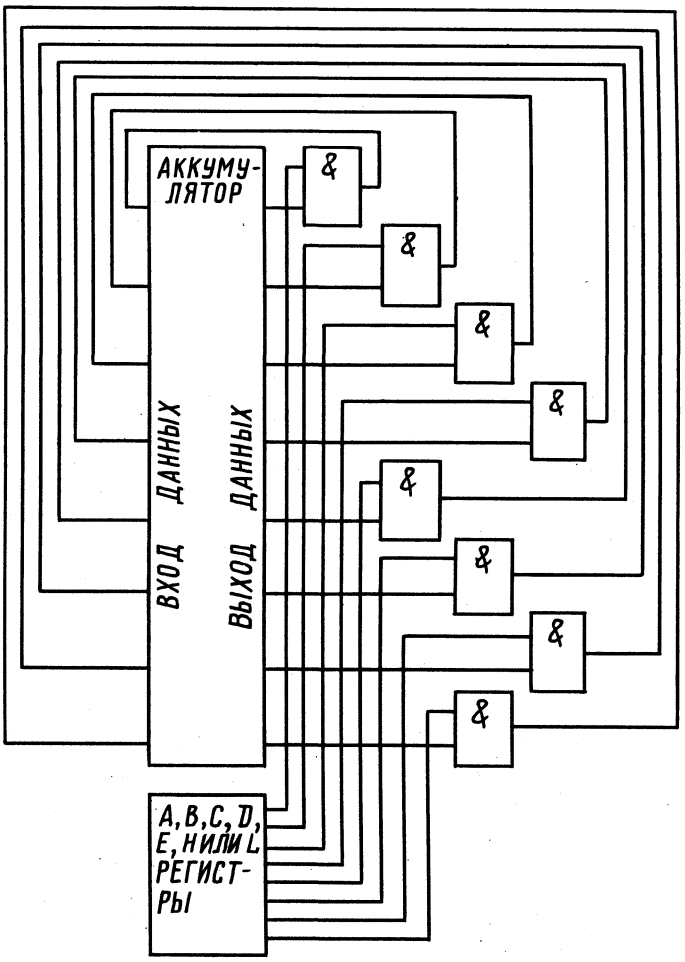


Рис.1. Аппаратный эквивалент команды

Для иллюстрации действия логических команд проведем следующий эксперимент: будем вводить данные с переключателей входного порта, затем эти данные преобразуются логическими командами, и результат высветится на светодиодных индикаторах выходного порта. Так как микролаборатория имеет только три переключателя входного порта, мы будем наблюдать только за тремя младшими разрядами байта данных /см.схему/.

1. Переведите программу /табл. 1/ в машинный код и введите в микролабораторию.

Таблица 1

Программа, демонстрирующая действие логических команд

Адрес	Содержимое	Метка	Команда	Комментарии
8000	...		MVI A, 81	Программирование интерфейса
8001	...			
8002	...		OUT FB	КР580ИК55
8003	...			
8004	...	START	IN FA	Чтение данных с входного порта в
8005				аккумулятор
8006	...		MVI B, OA	0000 0101 → B
8007	...			
8008	...		ANA B	A·B → A
8009	...		OUT F9	Запись данных в порт вывода
800A	...			
800B	...		JMP START	Зацикливание программы
800C	...			
800D	...			

Каждое мнемоническое обозначение должно быть преобразовано в соответствующий КОП и дополнено необходимыми данными /если нужно/. Эта программа считывает данные с переключателей входного порта, а затем объединяет их по "И" с числом 0000 1010. Результат появляется на светодиодных индикаторах. Затем программа возвращается на начало, и процесс бесконечно повторяется, так что можно изменить входные данные и проверить соответствующие выходные.

2. Проверьте правильность ввода программы.

3. Установите начальный адрес 8000 и нажмите кнопку ПУСК. Программа выполняется.

4. Установите входные переключатели в 1. Индикаторы показывают 0000 1010.

5. Измените положение переключателей и проверьте соответствующие выходы. Биты, которые объединяются по "И" с нулями, не зависят от положения переключателей.

6. Нажмите кнопку СБРОС для возвращения управления монитору. Заметьте, что переключатели не влияют больше на состояние индикаторов. Замените команду ANA B /по адресу 8008/ командой ORA B /КОП-B0/.

7. Запустите программу.

8. Посмотрите на соответствие между различными положениями входных переключателей и состояниями индикаторов. Заметьте различия между функциями "И" и "ИЛИ". Выходы битов, объединенных по "ИЛИ" с нулями, зависят от положения переключателей.

9. Нажмите кнопку СБРОС. Замените команду ORA B на XRA /КОП-A8/. Повторите шаги 7 и 8. Биты, объединенные по "Исключающему ИЛИ" с единицами, инвертируются.

МАСКИРОВАНИЕ

Типичное применение логических команд - выделение определенных битов слова /или маскирование/. Для примера предположим, что нам нужно проверить состояние одного из переключателей, подключенных к порту ввода, а состояние остальных проигнорировать. На рис.2 показана структурная схема программы, которая проверяет переключатель, соединенный со вторым битом входного порта. Если ключ замкнут, индикатор загорается и - наоборот.

Табл.2 показывает листинг программы. Сначала программа производит программирование интерфейса. Затем она считывает данные с входного порта в аккумулятор, засылает в регистр B маскирующее слово и объединяет по "И" регистр B и регистр A. В результате все биты, кроме второго, приводятся к нулю. Значение второго бита результата будет зависеть от положения переключателя. Программа использует команду JZ для перехода, если установлен флаг нуля.

Таблица 2

Программа проверки второго бита

Адрес	Содержимое	Метка	Команда	Комментарии
8000			MVI A,81	Программирование интерфейса
8001				
8002			OUT FB	
8003				
8004		START	IN FA	Чтение данных с переключателей
8005				в аккумулятор
8006			MVI B, 04	Запись в регистр B маскирующего
8007				слова /0000 0100/
8008			ANA B	A·B ⟶ A, выделение второго бита
8009			JZ OFF	Проверка аккумулятора на 0
800A				
800B				
800C		ON	MVI A, FF	Включение индикаторов
800D				
800E			OUT F9	
800F				
8010			JMP START	
8011				
8012				
8013		OFF	MVI A, 00	Выключение индикаторов
8014				
8015			OUT F9	
8016				
8017			JMP START	
8018				
8019				

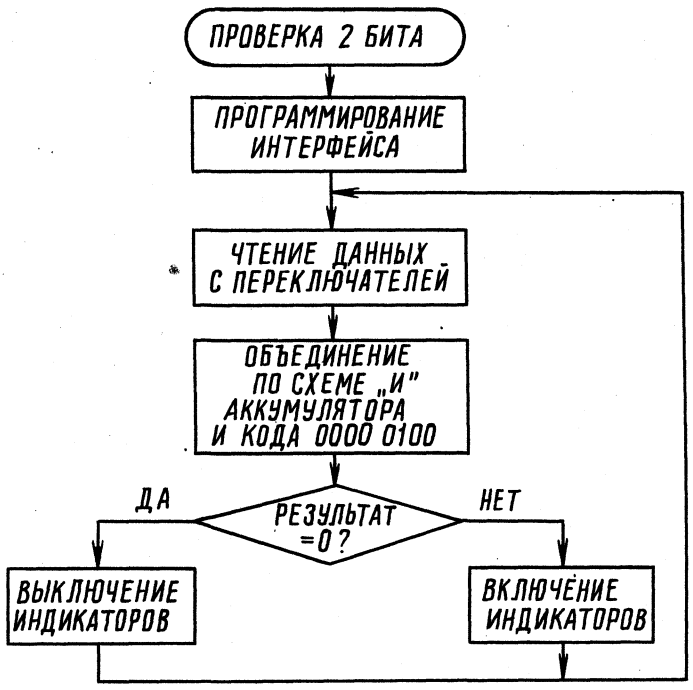


Рис.2. Структурная схема программы для проверки второго бита входного порта

Нулевой флаг индицирует, что полный байт /и поэтому бит 2/ равен 0.

Проведем эксперимент по выполнению этой программы.

1. Переведите программу в машинный код и введите в микролабораторию. Проверьте правильность ввода программы.

2. Запустите программу и проверьте, что входные переключатели правильно управляют индикаторами. Все индикаторы зажгутся только в том случае, если включен второй переключатель. Состояние остальных переключателей безразлично.

3. Измените программу, чтобы проверить другой переключатель. Если программа не работает, тщательно проверьте перевод в машинные коды и правильность ввода программы.

СБРОС АККУМУЛЯТОРА

Команда XRA A объединяет по "Исключающему ИЛИ" аккумулятор с

самим собой. Так как логическая функция "Исключающего ИЛИ" любого числа с самим собой дает в результате нуль, эта команда очищает аккумулятор, используя только один байт памяти. Другая команда MVI A, 0 требует два байта памяти.

СДВИГ ДАННЫХ

Другая часто необходимая функция - это сдвиг данных вправо или влево. В аппаратуре это выполняется с помощью сдвиговых регистров. МП КР580ИК80 имеет команды, которые сдвигают данные в аккумуляторе.

RRC /круговое вращение вправо/ производит сдвиг вправо, а L/C - влево. Эти команды оперируют только с аккумулятором. Круговое вращение означает, что младший бит сдвигается на место старшего /или наоборот/ /рис.3/.

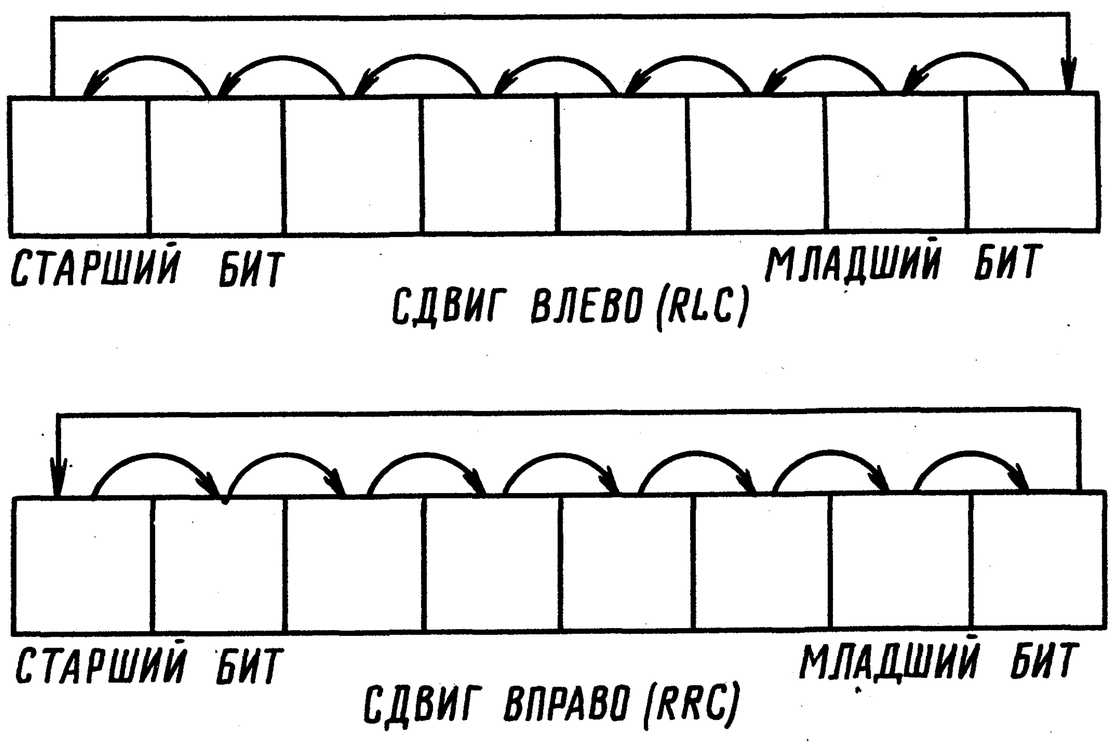


Рис.3. Функции команд сдвига

На рис.4 приведена структурная схема программы для демонстрации функций команд сдвига. При выполнении этой программы на индикаторах наблюдается свечение типа "бегущего огня".

Листинг программы приведен в табл.3. После программирования интерфейса программа заносит в аккумулятор код 1000 0000. Затем производится вывод содержимого аккумулятора на индикаторы. Команда вывода /OUT F9/ повторяется некоторое время, так как она находится в петле задержки, организованной с помощью регистров D и E. Задержка обеспечивает медленное изменение состояния индикаторов, без нее сдвиг данных аккумулятора происходил бы настолько быстро, что мы бы не могли проследить за ним.

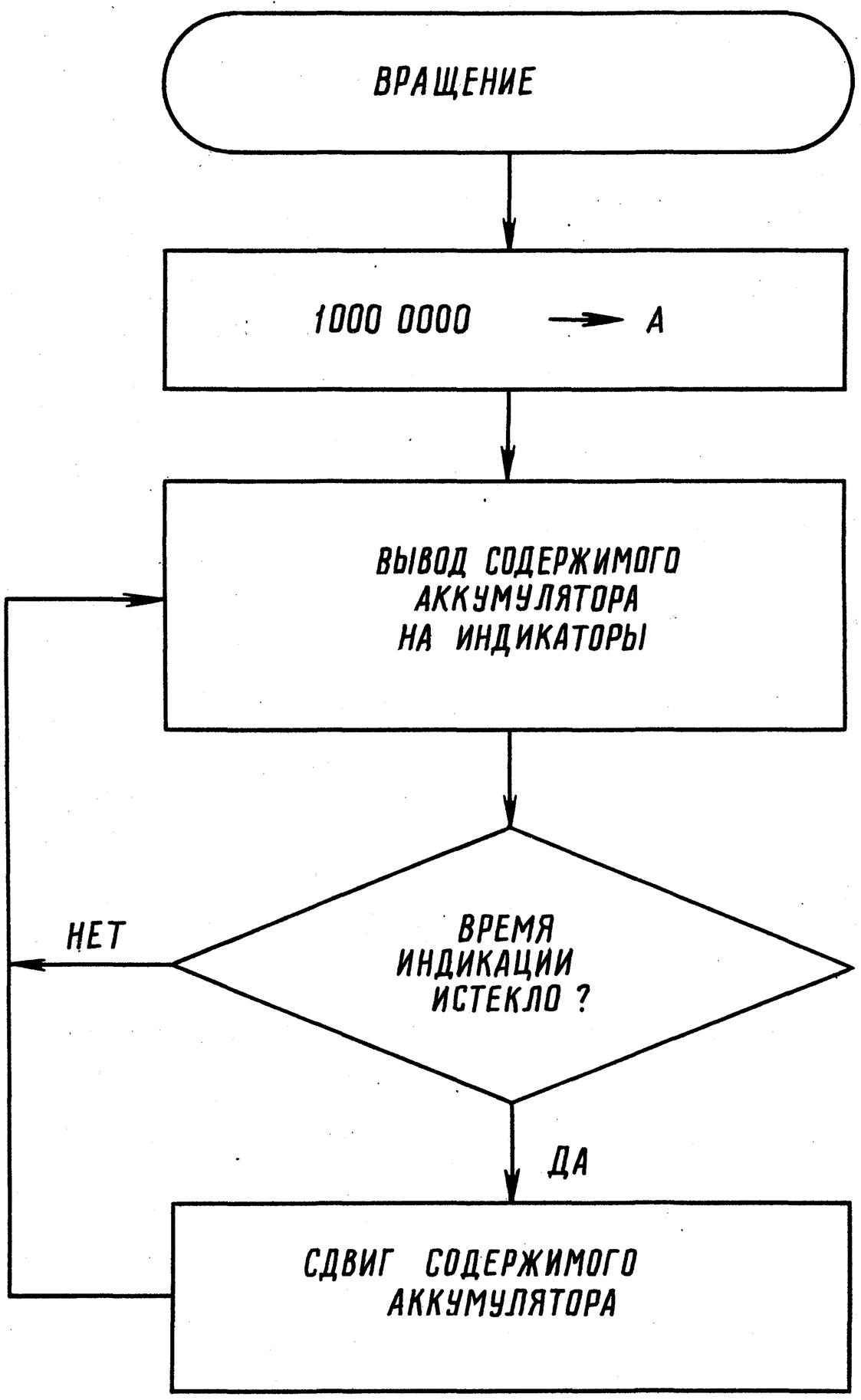


Рис.4. Структурная схема программы для демонстрации функций команд сдвига

Проведем эксперимент по выполнению этой программы.

1. Переведите программу в машинный код и введите в микролабораторию.

2. Проверьте правильность ввода программы и запустите ее. На индикаторах появится эффект "бегущего огня" вправо. Если не удалось достичь нужного эффекта, проверьте правильность машинных кодов записанной программы.

3. Замените команду RRC на RLC. В этом случае наблюдается "бегущий огонь" влево.

4. Вы можете также изменить частоту переключения индикаторов,

Таблица 3

Программа для демонстрации функций команд сдвига

Адрес	Содержимое	Метка	Команда	Комментарии
8000			MVI A,81	Программирование
8001				интерфейса
8002			OUT FB	
8003				
8004			MVI A,80	1000 0000 → A
8005				
8006		LOAD	MVI D,40	40 → D Загрузка регистров D
8007				и E для организации задержки
8008			MVI E, FF	FF → E
8009				
800A		LOAP	OUT F9	Вывод данных на индикаторы
800B				
800C			DCR E	E - 1 → E
800D			JNZ LOOP	Если E≠0, то переход на LOOP
800E				/петля задержки/
800F				
8010			DCR D	D - 1 → D
8011			JNZ LOOP	Если D≠0, то переход на LOOP
8012				/петля задержки/
8013			RRC	Сдвиг данных в аккумуляторе
8014			JMP LOAD	Повторение цикла
8015				
8016				
8017				

занося в регистр D различные числа /т.е. изменяя задержку/. Для этого достаточно менять код по адресу 8007, т.е. изменять второй байт команды MVI D. Заменяя код по адресу 8005, вы можете изменить комбинацию "зажженных" индикаторов.

АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Сложение

Основная арифметическая функция - это сложение двух чисел. Команда ADD R складывает данные определенного регистра и аккумулятора и запоминает результат в аккумуляторе. Например, ADD D складывает данные регистра D и аккумулятора. Положим D=1001 0011 и A=1010 1010, тогда результат будет:

в десятичной системе	в двоичной системе
147	1001 0011 D
+ 170	+ 1010 1010 A
317	10011 1101 A

Флаг переноса

Заметим, что в предыдущем примере результат больше 255_{10}, т.е. длиннее восьми бит. Для учета этого переполнения процессор содержит флаг переноса. Этот флаг работает как девятый бит аккумулятора и может быть проверен командами JC /переход, если есть перенос/ и JNC /переход, если нет переноса/. Эта проверка подобна проверке нулевого флага.

Например, программа ADD D; JC 8020 переводит содержимое регистра D в аккумулятор и переходит на ячейку 8020, если генерируется флаг переноса. Если переноса нет /результат меньше 256_{10}/, программа продолжается со следующей команды.

Флаг переноса применяется также для сложения чисел длиннее восьми бит. Два /или более/ регистра используются для хранения каждого числа. Сначала складываются младшие значащие байты, затем - старшие. Бит переноса выполняет функцию связи между двумя байтами.

Вы можете проверить флаги, заглянув в ячейку памяти 83EA. Флаги собраны в один байт, показанный на рис.5. В этом уроке используются только флаги нуля и переноса. Флаг знака копирует старший значащий бит /см.урок 15/. Не все команды изменяют флаги. Команда MOV, например, их не изменяет. Описание команд в приложении показывает, какие флаги генерируются каждой командой.

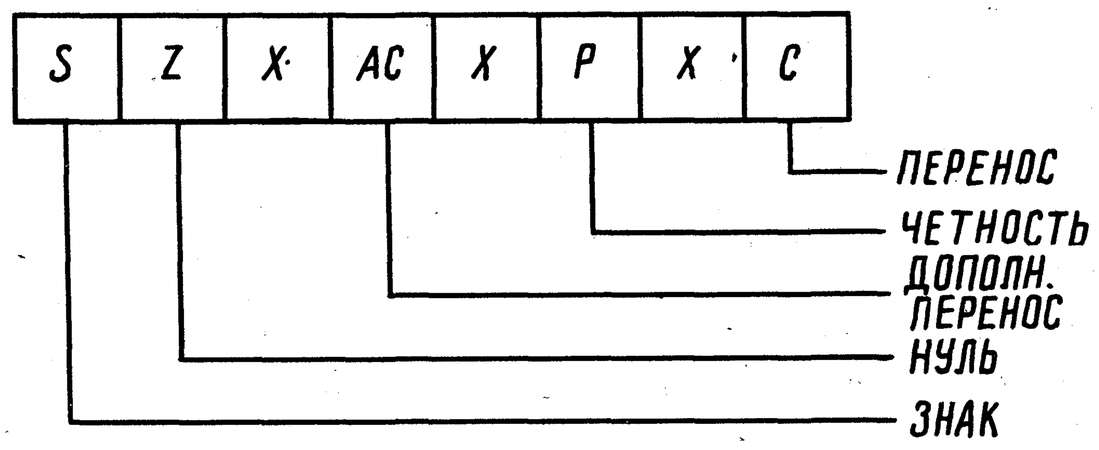


Рис.5. Флаги МП КР580ИК80

Вычитание

Команда SUB R вычитает содержимое определенного регистра из содержимого аккумулятора. Например SUB B вычитает из аккумулятора данные регистра B.

Команда вычитания использует флаг переноса как флаг заема. Если флаг переноса устанавливается после команды SUB R, значит число в регистре R больше, чем в A.

Проведем эксперимент по выполнению арифметических команд.

1. Дополните машинными кодами программу, приведенную в табл.4. Эта программа вычитает данные

Таблица 4

Программа демонстрации арифметических команд

Адрес	Содержимое	Метка	Команда	Комментарии
			SUB B	A-A → A
			ADD C	A+C → A

регистра B из аккумулятора, а затем прибавляет в аккумулятор данные регистра C.

2. Введите программу в микролабораторию. Перейдите в шаговый режим выполнения программ.

3. Занесите в регистр A число A7, в регистр B число 23, в регистр C число 93. Для этого необходимо устанавливать адрес ячейки памяти, соответствующей данному регистру и записывать туда необходимые данные /см. урок 5/. Напомним, что содержимое регистра A хранится в ячейке 83EB, регистра B - в ячейке 83E9, регистра C - в ячейке 83E8. При шаговом выполнении программы монитор записывает данные из этих ячеек в соответствующие регистры МП и после выполнения каждой команды записывает содержимое регистров обратно в соответствующие ячейки.

4. Установите начальный адрес программы и нажмите кнопку ПУСК. Команда SUB B должна выполниться. Проверьте содержимое аккумулятора, которое высвечивается цифровыми индикаторами. Оно должно быть равно $A7_{16}-23_{16}=84_{16}$. Проверьте содержимое флагового регистра. Флаги заема и переноса не установлены, так как результат не равен нулю и $23_{16} < A7_{16}$.

5. Нажмите кнопку ВОЗВР. Выполнилась команда ADD C. Содержимое аккумулятора становится равным $84_{16}+93_{16}=117_{16}$. Этот результат не вмещается в аккумулятор, в котором остается код 17_{16}, и формируется флаг переноса. Проверьте содержимое флага переноса и убедитесь, что он установлен.

6. Убедитесь, что при выполнении этой программы содержимое регистров B и C не изменилось.

Подпрограммы и стек

Большие программы могут быть упрощены путем применения подпрограмм для решения повторяющихся задач. Предположим, что программа должна производить серию вычислений, включающую в себя несколько умножений. В этом случае можно написать одну универсальную подпрограмму умножения, которую можно применять любое количество раз для выполнения умножения. Подпрограммы также удобны для деления программы на небольшие модули.

Для обращения к подпрограмме применяется команда CALL. Во время ее выполнения МП посылает содержимое программного счетчика /адрес

возврата/ в СТЕК - специально выбранную часть памяти. МП вернется к адресу возврата, когда выполнится команда RET в конце подпрограммы. После RET программа переходит к команде, непосредственно следующей за CALL.

Адрес возврата можно хранить в 16-разрядном регистре. Но в этом случае, если одна подпрограмма использует другую подпрограмму /вложение подпрограмм/, первый адрес возврата будет теряться.

Для решения этой проблемы и используется СТЕК. Когда одна программа вызывает другую, адреса возврата запоминаются в последовательных ячейках памяти. Рис.6. показывает последовательность действий, когда программа А вызывает программу В, которая вызывает программу С. Содержимое программного счетчика и СТЕКА в каждом шаге показано на рис.7. Последовательность следующая:

1 - выполняется программа А;

2 - программа А обращается к подпрограмме В. Адрес возврата запоминается в СТЕКЕ, а программный счетчик загружается стартовым адресом программы В;

3 - программа В обращается к программе С. Адрес возврата для программы В помещается в СТЕК, а в программный счетчик загружается стартовый адрес программы С;

4 - возврат из программы С. Последняя строка СТЕКА загружается в программный счетчик, и управление возвращается к программе В;

5 - возврат из программы В к программе А. МП КР580ИК80 использует ОЗУ в качестве СТЕКА, емкость которого неограничена. Специальный регистр процессора - указатель СТЕКА /SP/ содержит адрес вершины СТЕКА. Область СТЕКА в ОЗУ определяется при занесении в указатель СТЕКА нужного адреса.

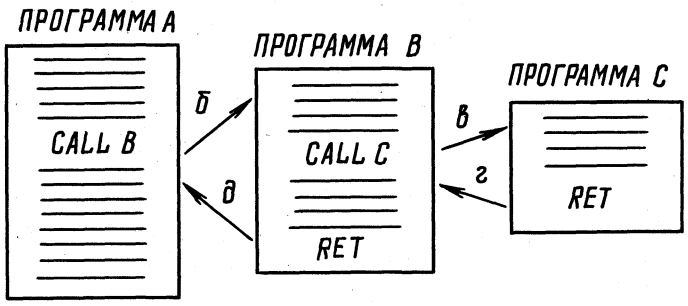


Рис.6. Вложение подпрограммы

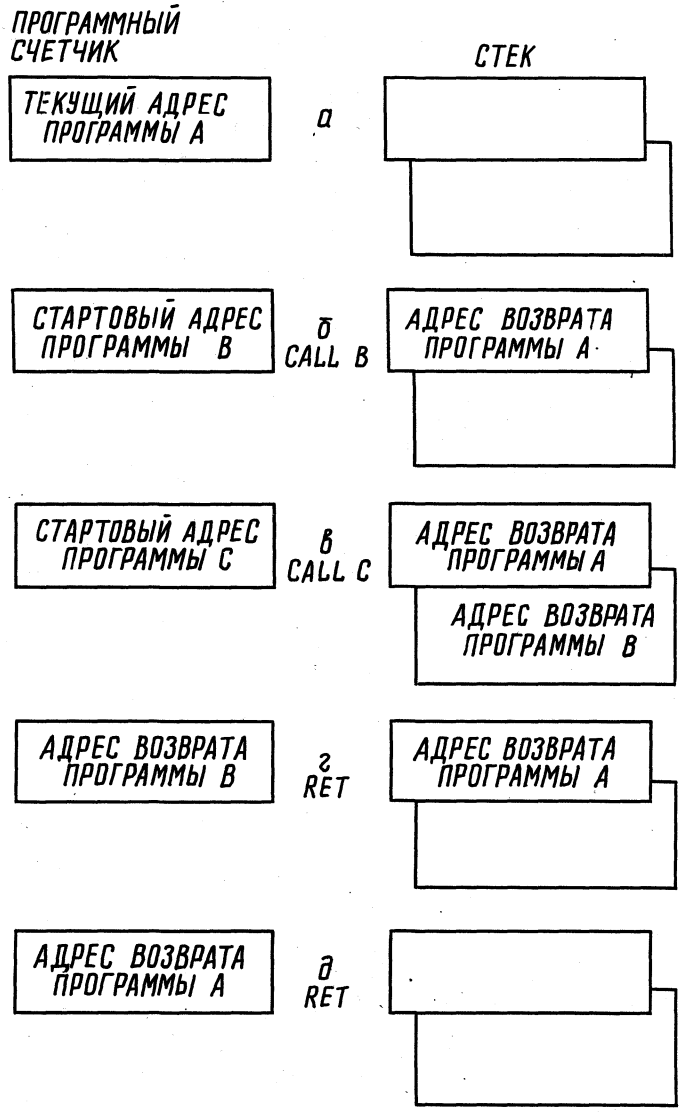


Рис.7. Операции со СТЕКОМ

Монитор микролаборатории при включении питания и после нажатия кнопки СБРОС заносит в указатель СТЕКА число 83С7. Это позволяет применять СТЕК без предварительной установки указателя СТЕКА. Для программы возможна самостоятельная установка указателя СТЕКА, если это требуется. Однако, когда вы в следующий раз нажмете СБРОС, монитор установит содержимое указателя СТЕКА в 83С7.

Можно проверить указатель СТЕКА так же, как и регистры МП. Указатель СТЕКА находится в ячейке 83Е3 /старший байт/ и 83Е2 /младший байт/.

Команды PUSH и POP

Команды CALL и RET оперируют указателем СТЕКА автоматически. Существуют команды, позволяющие ис-

пользовать СТЕК по вашему желанию. Команда PUSH засылает данные пары регистров в СТЕК, а команда POP считывает данные назад в регистры. Например, PUSH B проталкивает в СТЕК данные регистра B, а затем регистра C. Команда POP B возвращает оба эти регистра с вершины СТЕКА.

Эти команды совместно применяются для временного хранения содержимого регистров. Допустим, что программа использует B и C регистры для запоминания каких-то данных. Она вызывает подпрограмму, которая также использует эти регистры. Подпрограмма может сохранить первоначальное содержимое этих регистров /как предустановку при вызове подпрограммы/ при использовании PUSH B. Затем, перед окончанием подпрограммы команда POP B используется для возвращения регистрам B и C их первоначальных значений. Запоминание содержимого этих регистров может осуществить и главная программа, если перед CALL выполнить команду PUSH B, а после CALL - POP B.

МЕТОДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

СИСТЕМА РАЗРАБОТКИ

При написании программ с использованием "Микролаб" нужно вручную переводить язык Ассемблер на машинный язык. Затем машинный код с клавиатуры вводится в "Микролаб". Эта система подходит для написания коротких программ, но она слишком трудоемка для больших программ.

Большая часть программного обеспечения МП написана с использованием системы разработки, т.е. специально предназначенной для этого микро-ЭВМ. В типичную систему входит ОЗУ емкостью 32-64К байт, дисководы гибких дисков с двойной плотностью записи для хранения программ, терминал ЭЛТ с клавиатурой и печатающее устройство. Программное обеспечение системы разработки включает редактор для ввода и редактирования программ на языке Ассемблер, Ассемблер для перевода на машинный язык и монитор или отладчик для установки точек прерывания, проверки регистров и т.д. Такая система значительно облегчает процесс разработки.

Для облегчения отладки программного обеспечения многие системы разработки включают внутрисхемный эмулятор /ICE/. МП вынимается из испытуемой системы, а на его место вставляется кабель из системы разработки с 40-выводным разъемом на конце. Внутрисхемный эмулятор - одно из наиболее сложных отладочных устройств.

ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При разработке сложных программ необходимо придерживаться следующей последовательности:

определить задачу;

принять решение - разделить на функциональные блоки;

составить алгоритм программы;

написать программу;

испытать и отладить каждую команду;

испытать и отладить полную программу.

Первый шаг очевиден. До того, как принять решение, необходимо понять задачу. Нужно составить набор характеристик программы: определить, что является входными параметрами, как их нужно обработать, что является выходом.

После определения этих характеристик можно приступить к разработке программы. Сначала надо продумать ее, общую конструкцию. Лучше разделить программу на небольшие модули, каждый из которых должен иметь четко определенные вход и выход. Поиск дефектов в программе, разделенной на функциональные блоки, значительно облегчается. Такой же метод применяется при конструировании аппаратной части.

Одним из методов определения структуры разрабатываемой программы является структурное программирование, общий принцип которого приводится в данном разделе.

После определения задачи, выбора метода и разделения программы на модули можно приступить к составлению алгоритмов. Обычно составляется общий алгоритм, в котором каждый

модуль программы изображается в виде квадрата. Составляются алгоритмы и для каждого модуля.

После этого можно писать программу. Тщательность выполнения предшествующих шагов облегчает написание программы.

После составления программы ее необходимо проверить. Модульность программы облегчает проверку. Каждый модуль может быть проверен отдельно. Когда все модули проверены, можно соединить их вместе и проверять всю программу.

Пример разработки программного обеспечения

Рассмотрим описание программы контроллера "бегущие огни".

Индикаторы выходного порта /светодиоды/ используются как имитаторы "бегущих огней". Контроллер должен регулировать следующую последовательность чередования горения светодиодов:

1. Горят светодиоды 1, 4, 7; остальные погашены.
2. Время горения установленных светодиодов.
3. Горят светодиоды 2, 5, 8; остальные погашены.
4. Время горения установленных светодиодов.
5. Горят светодиоды 3, 6; остальные погашены.
6. Время горения установленных светодиодов.
7. Повторение процесса /переход к шагу 1/.

На рис.1. приведен алгоритм работы контроллера.

Выходные индикаторы /светодиоды YDI...YD8/ подключаются к шине данных с помощью программируемого интерфейса.

В табл.1 показаны используемые комбинации сигналов имитирования "бегущих огней" на светодиодах. Единица в двоичном коде сигнала означает включенный индикатор.

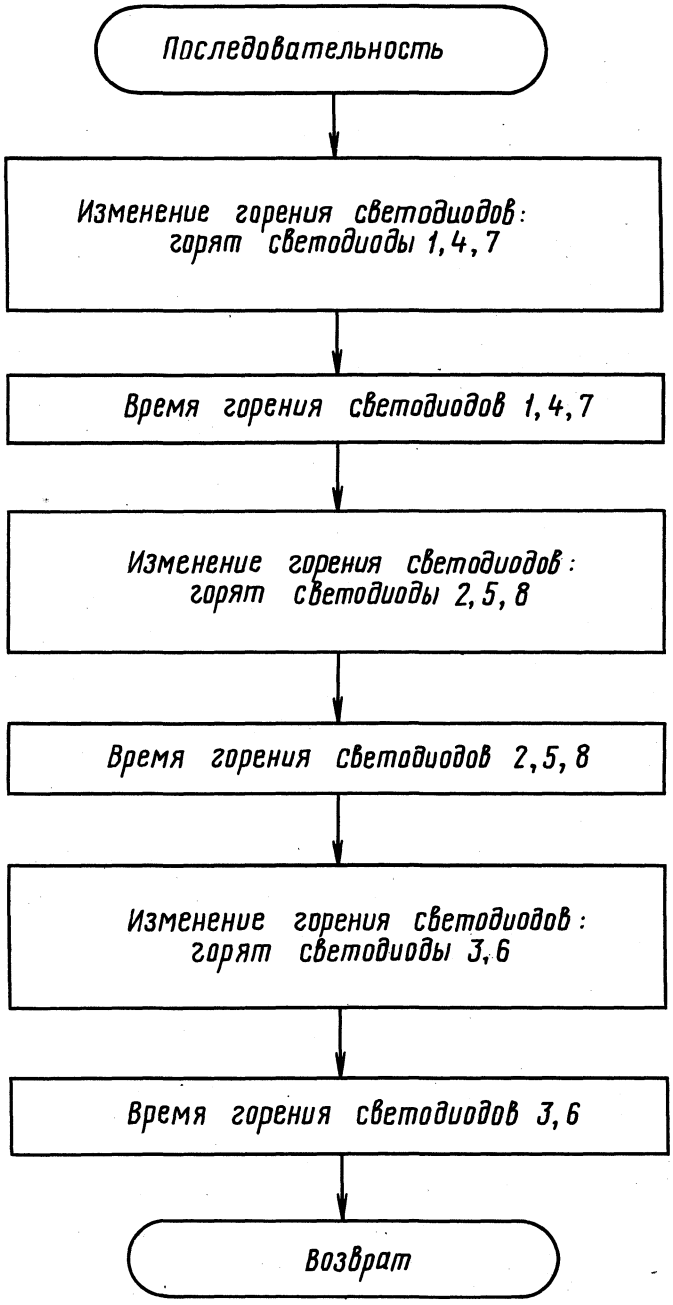


Рис.1. Алгоритм программы последовательности горения светодиодов

Таблица 1

VD1	VD2	VD3	VD4	VD5	VD6	VD7	VD8	Код	
1	0	0	1	0	0	1	0	92	16
0	1	0	0	1	0	0	1	49	16
0	0	1	0	0	1	0	0	24	16

Программа задержки

Для работы основной программы необходимо разработать подпрограмму, вызывающую задержку. Простейший способ генерирования задержки показан на рис.2. Однако, максимальное время задержки, полученное таким способом, не превышает 1785 мкс. Принцип получения более длительной задержки будет описан ниже.

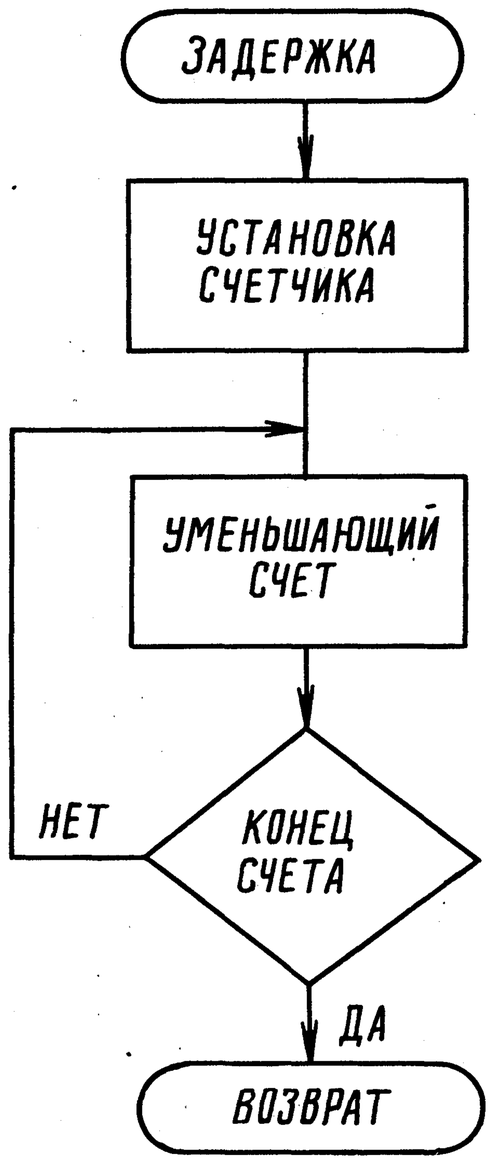


Рис.2. Алгоритм программы задержки

ПРОГРАММА КОНТРОЛЛЕРА "БЕГУЩИЕ ОГНИ"

После составления алгоритмов и определения характеристик, можно писать программы. Основная программа последовательности довольно проста /табл.2/.

Таблица 2

Метки	Команды	Комментарии
SEQ	MOV A, 92	Установка горения светодиодов 1,4,7
	MVI D, I	Время горения установленных сигналов
	CALL DELAY	
	MOV A, 49	Установка горения светодиодов 2,5,8
	MVI D, I	Время горения установленных сигналов
	CALL DELAY	
	MOV A, 24	Установка горения светодиодов 3,6
	MVI D, I	Время горения установленных сигналов
	CALL DELAY	

Программа последовательности устанавливает в регистр A число 92 - код выходных индикаторов, при котором светодиоды 1, 4, 7 - горят, а светодиоды 2, 3, 5, 6, 8 - погашены.

Затем вызывает подпрограмму задержки. Для чего программа последовательности задает величину задержки в регистр D, после чего вызывается сама подпрограмма задержки. Далее эта последовательность повторяется для других кодов выходных сигналов светодиодов.

РАЗРАБОТКА ПРОГРАММЫ ЗАДЕРЖКИ

В табл.3 приведена программа задержки, алгоритм которой показан на рис.2.

Таблица 3

Простая программа задержки

Метки	Команды	Комментарии	Примечание
DELAY	DCR A	Уменьшающий счет	4 такта
	JNZ DELAY	Продолжение счета, если содержимое аккумулятора 0	7/10 тактов
	RET	Возврат	10 тактов

Чтобы использовать эту программу, в аккумуляторе устанавливается нужное значение задержки и вызывается программа задержки. Содержимое аккумулятора уменьшается до тех пор, пока не достигает нуля, после чего управление возвращается на программу вызова.

Время задержки, полученное при выполнении этой программы, можно проанализировать следующим образом:

в примечаниях указано число периодов синхроимпульсов /тактов/, необходимых для выполнения каждой команды;

для DCR A требуется четыре такта. Для JNZ, в случае выполнения условия, требуется 10 тактов, если же условие не выполняется - 7 тактов. Для RET требуется 10 тактов. Для каждого прогона по петле требуется 4+10=14 тактов. Для последнего прогона требуется 4+7+10=21 такт, так как команда JNZ не вызывает перехода и требует семь тактов, для команды RET требуется 10 тактов.

Следовательно, уравнение для задержки имеет вид:

задержка=/A-1/·14+21 /тактов/

Один такт длится 0,5 мкс. Поэтому уравнение можно переписать следующим образом:

задержка=/A-1/·7 мкс+10,5 мкс

Здесь A - начальное содержимое аккумулятора.

Самая длинная задержка вызывается установкой аккумулятора в нуль. При первом прогоне по петле содержимое аккумулятора уменьшается и становится равным 255. Первоначальная установка в нуль эквивалентна тому, что A=256. Следовательно, максимальная задержка равна /256-1/·7+ +10,5=1795 мкс.

Использование пар регистров

Для получения более длительной задержки целесообразно применить команды, воздействующие на пары регистров.

Программа, использующая пары регистров, приведена в табл.4. Использование команды DCX B, уменьшающей содержимое пары регистров B, C позволяет увеличить счет задержки до значения 256^2=65536. Так как команда DCX B не влияет на нулевой флаг МП, то для этого используется команда OARC. После этого может быть использована команда JNZ.

Таблица 4

Метки	Команды	Комментарии
DELAY	DCX B	Уменьшающий счет
	MOV A, B	Проверка на 0
	ORA C	
	JNZ DELAY	Повторить до 0
	RET	

Время задержки, полученное при выполнении этой программы, определяется по формуле, имеющей вид:

задержка=/N-1/·24+31 /тактов/=
=/N-1/·12 мкс+15,5 мкс .

Здесь N - начальное содержимое регистров B, C, максимальная задержка=/65536-1/·12+ +15,5=0,786 с.

Такую задержку можно получить, если первоначально занести 0 в регистры B, C.

Для получения задержки, используемой в программе контроллера "бегущие огни", организуется счетчик в регистре D. Число, первоначально записанное в регистр D, определяет, сколько раз выполняется задержка 0,786 с. Максимальная задержка, полученная таким образом, равна 256x0,786=201 с.

В табл.5 приведена программа задержки, используемая для контроллера "бегущие огни".

Таблица 5

Метки	Команды	Комментарии
DELAY	LXI B, 0	Начало счета внутренней петли
LOOP	DCX B	Петля задержки 0,786 с
	MOV A,Б	
	ORA C	
	JNZ LOOP	
	DCR D	Уменьшение основного счетчика и работа по петле, если ≠0
	RET	

Команда LXI B, 0 заносит 0 в пару регистров B, C. Далее вырабатывается задержка 0,786 с. Регистр D используется для счета основной

задержки. В этой программе две петли: внешняя и внутренняя. Внутренняя петля генерирует задержку 0,786 с, внешняя - множество таких задержек. Такая организация программы называется вложением.

Полная программа контроллера "бегущие огни" приведена в табл.6.

Таблица 6

Адреса	Коды	Метки	Команды	Комментарии
8000	3E	MOV A,81	MOV A,81	Программирование интерфейса
8001	81			
8002	D3			
8003	FB		OUT FB	
8004	3E	SEQ	MOV A,92	Установка кода горения 1,4,7 светодиодов
8005	92			
8006	D3		OUT F9	
8007	F9			
8008	16		MVI D, I	Горение данной последовательности
8009	01			
800A	CD		CALL DELAY	
800B	22			
800C	80			
800D	3E		MOV A,49	Установка кода горения 2,5,8 светодиодов
800E	49			
800F	D3		OUT F9	
8010	F9			
8011	16		MVI D, I	Горение данной последовательности
8012	01			
8013	CD		CALL DELAY	
8014	22			
8015	80			
8016	3E		MOV A, 24	Установка кода горения 3,6 светодиодов
8017	24			
8018	D3		OUT F9	
8019	F9			
801A	16		MVI D,I	Горение данной последовательности
801B	01			
801C	CD		CALL DELAY	
801D	22			
801E	80			
801F	C3		JMP SEQ	Возврат в основную программу "бегущие огни"
8020	04			
8021	80			
8022	01	DELAY	LXI B,0025	Начало внутренней петли задержки
8023	00			
8024	25			
8025	OB	LOOP	DCX B	Внутренняя петля, генерирующая задержку 0,786 с
8026	78		MOV A,B	
8027	B1		ORA C	
8028	C2		JNZ LOOP	

Продолжение табл.6

Адреса	Коды	Метки	Команды	Комментарии
8029	25			
802A	80			
802B	15		DCR D	Основная петля задержки
802C	C2		JNZ DELAY	
802D	22			
802E	80			
802F	C9		RET	Возврат

Проверка программы

1. Введите программу из табл.6 в ОЗУ аккумулятора.

2. Удостоверьтесь, что она правильно записана в память.

3. Выполняйте основную программу, начинающуюся по адресу 8000.

4. Значения временных задержек можно заменить на другие значения, для уменьшения скорости "бегущих огней" по адресам 8009, 8012,801A, 8023, 8024.

Урок 14

КЛАВИАТУРА И ИНДИКАЦИЯ

КЛАВИАТУРА

Клавиатура представляет собой набор ключей /кнопок/, организованных в виде матрицы 8x3.Каждый ряд из 8 кнопок опрашивается отдельно /аппаратное обеспечение описано в уроке 9/. Считываемые данные преобразуются в код, соответствующий нажатой кнопке, с помощью программы монитора KEYIN /адрес 0216H/. Эта программа вызывается с помощью команды CALL KEYIN. После возврата из программы KEYIN в аккумуляторе будет содержаться код нажатой кнопки /табл.1/. Эта мониторная программа может использоваться для чтения клавиатуры.

Таблица 1

Коды кнопок микролаборатории

Кнопка	Код /H/
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
A	OA
B	OB
C	OC
D	OD
E	OE
F	OF
ПУСК	10
ВОЗВР.	11
УСТ.АД.	12
АД.-	13
АД.+	14
ЗП	15
ВЫВОД	16
ВВОД	17

Продолжение табл.1

В табл.2 показан листинг программы, которая вызывает звуковой сигнал, если нажата кнопка 7. Клавиатура читается с помощью вышеописанной программы KEYIN, которая заносит код нажатой клавиши в аккумулятор.

Команда CPI 07 устанавливает нулевой флаг процессора, если содержимое аккумулятора равно 07. Команда JNZ READ вызывает возврат программы на начало, если нулевой флаг не установлен. Если нулевой флаг установлен, то будет вызываться подпрограмма BEEP. Затем процесс повторяется. Таким образом, звуковой сигнал будет генерироваться при нажатии кнопки 7.

Введите листинг программы в микролабораторию и проверьте пра-

Таблица 2

Программа вызова звукового сигнала

Адрес	Содержимое	Метка	Команда	Комментарии
8000	CD	READ	CALL KEYIN	Вызов программы, чтение с
8001	16			клавиатуры
8002	02			
8003	FE		CPI 07	Сравнение кода ключа с 07
8004	07			
8005	C2		JNZ READ	Возврат, если кнопка 7 не
8006	00			нажата
8007	80			
8008	CD		CALL BEEP	Звуковой сигнал, если кнопка
8009	50			7 нажата
800A	03			
800B	C3		JMP READ	Повторение программы
800C	00			
800D	80			

вильность ввода. Запустите программу.

Нажмите кнопку 7 - динамик гудит. Нажмите любую другую кнопку - звуковой сигнал отсутствует.

Составьте программу, которая генерирует звуковой сигнал, если нажаты в определенной последовательности три определенные кнопки, например 7, 3, 9. Эту программу можно взять за основу при создании электронного замка. Составление программы начните со структурной схемы. Введите программу в микролабораторию и проверьте ее выполнение.

СКАНИРОВАНИЕ КЛАВИАТУРЫ

На рис.1 приведена упрощенная схема клавиатуры. За один цикл считываются данные с одного ряда из восьми кнопок. Для простоты рассмотрим только один ряд, т.е. кнопки 0, 1, 2...7.

Чтение ряда ключей представляет собой двухшаговую операцию: запись данных в сканирующий порт для выбора кнопок, чтение байта данных читающим портом.

Чтобы выбрать определенный ряд кнопок, соответствующий бит сканирующего порта /PC4, PC5 или PC6/ устанавливается в лог.1, другие биты остаются равными лог.0. Таким образом, чтобы выбрать ряд кнопок 0...7, нужно послать в порт FA код 1001 1111 /9F-H/, чтобы выбрать кнопки 8...F - код 1010 1111 /AFH/ и чтобы выбрать кнопки ПУСК...ВВОД,- код 1100 1111 /CFH/. Биты PC0...PC3, PC7 не используются при сканировании клавиатуры и устанавливаются равными лог.1.

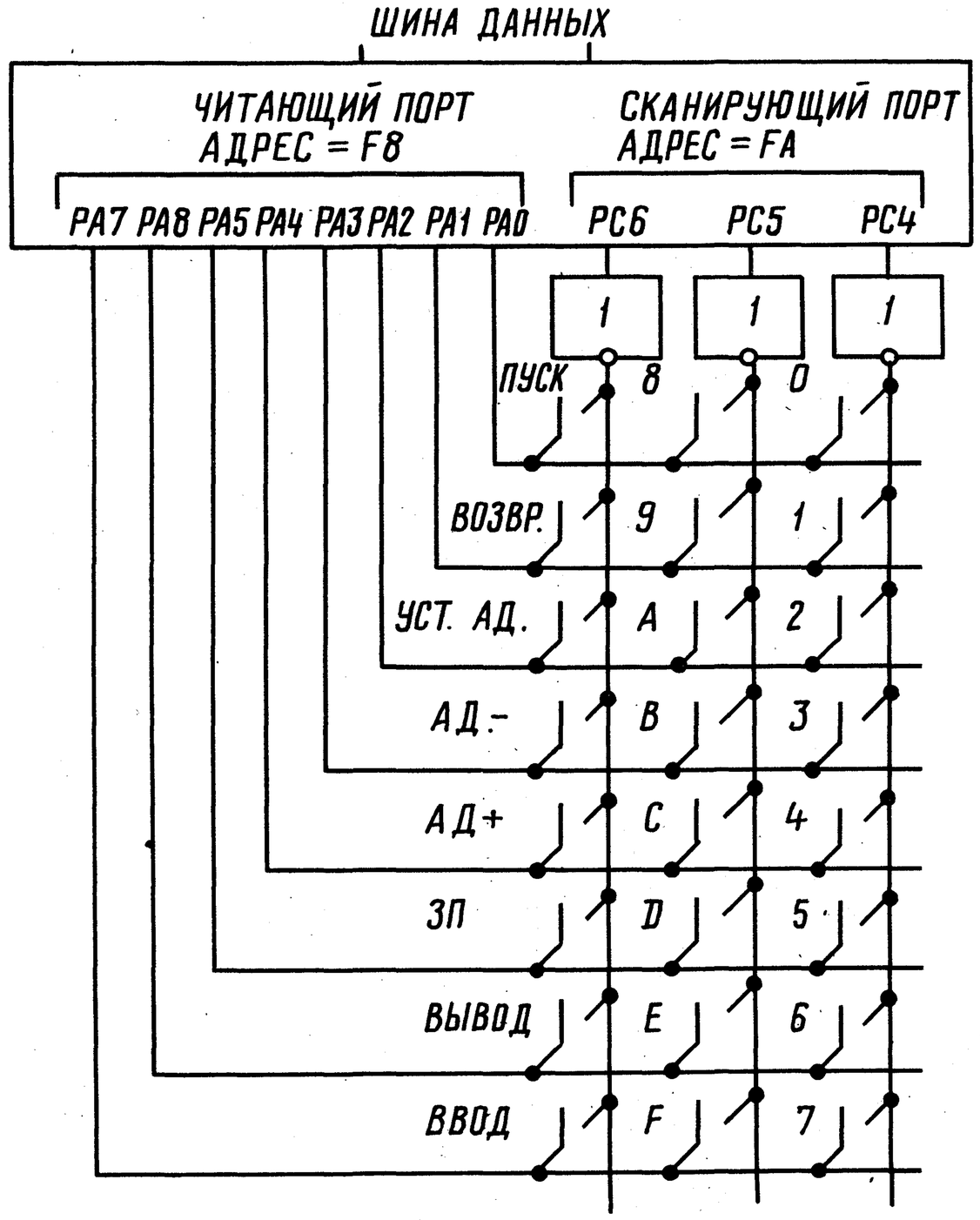


Рис.1. Упрощенная схема клавиатуры

Итак, чтобы считать данные с ряда 0...7, в порт FA выводится байт 9F. Затем производится чтение входным портом для сбора информации о состоянии колонок. При нажатии определенной кнопки соответствующий бит читающего порта будет устанавливаться в 0 /табл.3/. /Обратите внимание, что коды кнопок, приведенные в табл. 1 и 3, не совпадают/. Кнопка из других рядов при чтении выбранного ряда не влияет на результаты чтения. Табл.4 показывает программу, реализующую только что описанный процесс чтения.

Таблица 3

Коды, читаемые с клавиатуры

Кнопка	Читаемый код /двоичный/	Читаемый код /шестнадцатеричный/
Нет нажатых кнопок	1 1 1 1 1 1 1 1	FF
0	1 1 1 1 1 1 1 0	FE
1	1 1 1 1 1 1 0 1	FD
2	1 1 1 1 1 0 1 1	FB
3	1 1 1 1 0 1 1 1	F7
4	1 1 1 0 1 1 1 1	EF
5	1 1 0 1 1 1 1 1	DF
6	1 0 1 1 1 1 1 1	BF
7	0 1 1 1 1 1 1 1	7F

Таблица 4

Программа определения состояния кнопок

Команда	Комментарии
MVI A, 9F OUT FA	Выбор ряда кнопок 0...7
IN F8	Чтение состояния кнопок

Программа KEYIN, описанная выше, не только сканирует клавиатуру, но и преобразует коды, читаемые с колонок /в зависимости от выбранного ряда/, в коды, приведенные в табл.1.

В табл.5 приведена программа для проверки состояния кнопки 2, которая использует вышеописанный способ чтения клавиатуры. Если кнопка 2 нажата, генерируется зву-

Таблица 5

Программа проверки состояния кнопки 2

Адрес	Содержимое	Метка	Команда	Комментарии
8000	3E		MVI A,9F	Выбор ряда кнопок 0...7
8001	9F			
8002	D3		OUT FA	
8003	FA			
8004	DB	READ	IN F8	Чтение состояния колонок
8005	F8			
8006	FE		CPI FB	Сравнение аккумулятора с с кодом /кнопка 2/
8007	FB		JNZ READ	Читать снова, если кнопка
8008	C2			2 не нажата
8009	04			
800A	80			
800B	CD		CALL BEEP	Если нажата кнопка 2,
800C	50			появляется звуковой сигнал
800D	03			
800E	C3		JMP READ	Повторение программы
800F	04			
8010	80			

ковой сигнал. Наберите программу на "Микролаб" и проверьте ее работу.

УСТРАНЕНИЕ ЗВОНА

Другим фактором, который должен быть рассмотрен при чтении с клавиатуры, является исключение "звона". Как известно, кнопки при нажатии не выдают "чистый" импульс - они "звенят", как показано на рис.2. Когда металлические контакты касаются один другого, возникает звон. Продолжительность звона 1-50 мс в зависимости от конструкции кнопки. Звон контактов нужно учитывать, так как он может привести к тому, что вместо одного нажатия будет зафиксировано много нажатий кнопки.

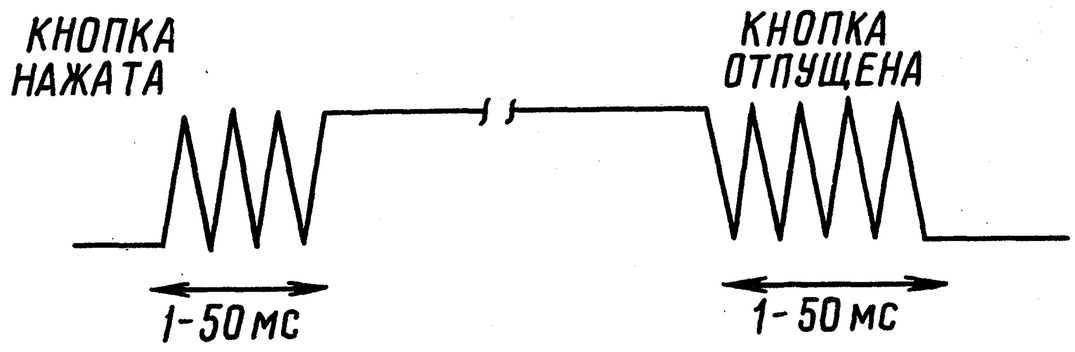


Рис.2. Звон кнопки

Звон можно исключить путем введения схемы исключения звона в читающий порт или с помощью специальной подпрограммы.

Рис.3. показывает один из путей исключения звона с помощью программы. Клавиатура сканируется, используя программу KEYIN, описанную ранее. Если нет нажатых кнопок, процесс повторяется. Когда же определено нажатие кнопки, клавиатура отключается на 50 мс, чтобы не читать во время звона. Затем клавиатура сканируется до момента, пока ключ не отпущен. Опять вводится 50 мс задержки, чтобы исключить звон при отпускании кнопки.

ИНДИКАЦИЯ

В микролаборатории для индикации используются восемь семисегментных цифровых индикаторов.

Каждому индикатору соответствует ячейка памяти, где хранится семисегментный код, управляющий свечением сегментов индикатора /см.урок 9/. Информация из этих

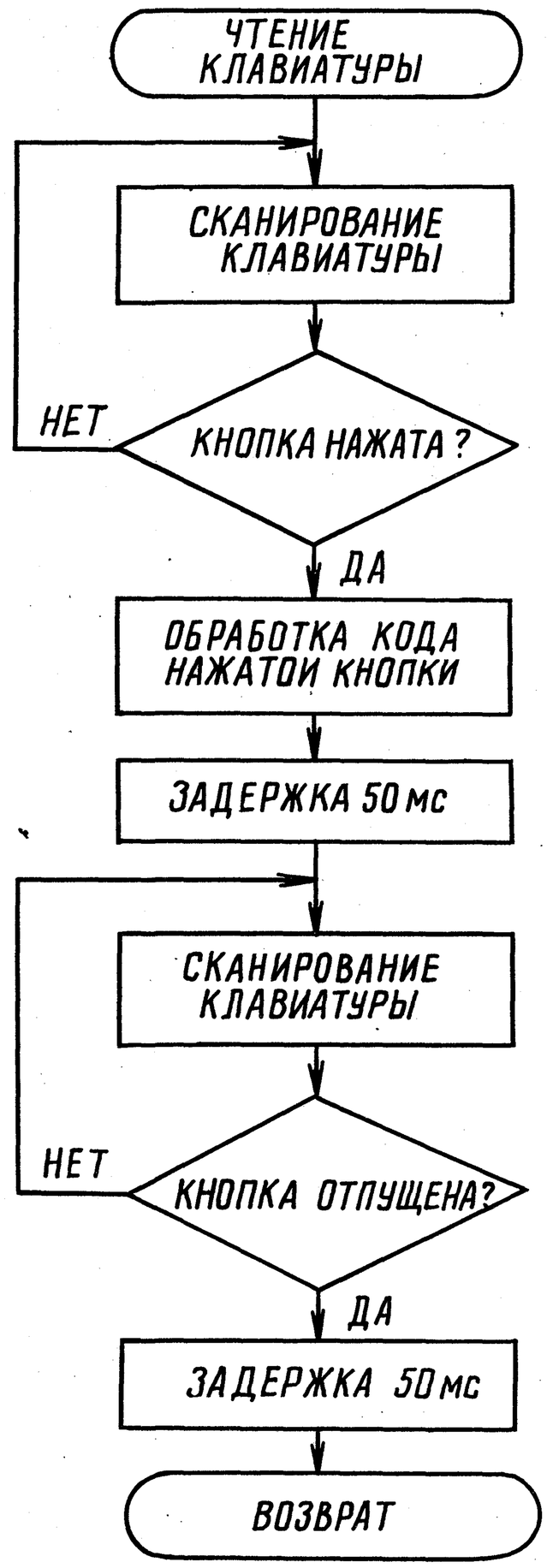


Рис.3. Алгоритм устранения звона клавиатуры

ячеек посылается на индикаторы при помощи специальной схемы, которая обеспечивает динамический режим индикации.

Задача матобеспечения сводится к подготовке семисегментных кодов и засылке их в ячейки памяти 83F8...83FF. Ячейка 83F8 соответствует левому индикатору, 83FF - правому. /Динамическая индикация осуществляется в микропроцессорных системах не обязательно с помощью схемы. Сканирование индикаторов можно осуществлять и с помощью программы/.

Монитор микролаборатории содержит программу SEGCG /адрес=01C0/, которая преобразует шестнадцатеричные коды 0, 1, 2...4, E, F в семисегментные и заносит их в ячейки памяти 83F8...83FF. Исходные данные для этой программы должны находиться в

ячейках 83F4...83F7. Каждый байт из этих ячеек соответствует паре индикаторов: содержимое ячейки 83F4 после преобразования высветится на двух левых индикаторах, содержимое ячейки 83F7 - на двух правых.

В табл.6 приведена программа показа информации, использующая подпрограмму SEGCG. Введите программу и проверьте ее работу. На индикаторах должны последовательно высвечиваться символы 1, 2, 3, 4, A, B, C,D.

Таблица 6

Программа показа информации

Адрес	Содержимое	Метки	Команда	Комментарии
8000	3E		MVI A,12	12→[A]
8001	12			A→[83F4]
8002	32		STA 83F4	
8003	F4			
8004	83			
8005	3E		MVI A,34	34→A
8006	34			
8007	32		STA 83F	A→[83F5]
8008	F5			
8009	83			
800A	3E		MVI A,AB	AB → A
800B	AB			
800C	32		STA 83F6	A→[83F6]
800D	F6			
800E	83			
800F	3E		MVI A,CD	CD → A
8010	CD			
8011	32		STA 83F7	A→[83F7]
8012	F7			
8013	83			
8014	CD		CALL SEGCG	Вызов SEGCG
8015	CO			
8016	01			
8017	76		HLT	Останов

Напишите программу,выводящую на два правых индикатора значение нажатой кнопки, а на остальные индикаторы - нули. Используйте при этом мониторные подпрограммы KEYIN и SEGCG.

В предыдущей программе мы использовали подпрограмму SEGCG для получения семисегментного кода. Но эта подпрограмма оперирует только с шестнадцатеричными символами - 0...F, т.е. не полностью использует возможности семисегментных индикаторов.

Можно формировать семисегментный код и получать на индикаторах любые символы, которые позволяют получить используемые в микролаборатории индикаторы. Как уже говорилось, семисегментные коды должны засылаться в определенные ячейки памяти. Каждый бит в этих ячейках соответствует определенному сегменту /рис.4/. Если бит равен 1, то соответствующий сегмент будет "гореть", и - наоборот.

БИТ	7	6	5	4	3	2	1	0
СЕГМЕНТ	H	G	F	E	D	C	B	A

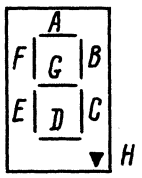


Рис.4. Соответствие разрядов и сегментов

В табл.7 приведен листинг программы, которая выводит на индикаторы надпись "СССР 60". Изменяя семисегментные коды, вы можете вывести на индикацию любую возможную надпись.

Таблица 7

Программа вывода символов на индикаторы без подпрограммы SEGCG

Адрес	Содержимое	Метка	Команда	Комментарии
8000	3E		MVI A, 39	Семисегментный код буквы
8001	39			C → A
8002	32		STA 83F8	A → [83F8]
8003	F8			
8004	83			
8005	32		STA 83F9	A → [83F9]
8006	F9			
8007	83			

Продолжение табл.7

Адрес	Содержимое	Метка	Команда	Комментарии
8008	32		STA 83FA	A ⟶ [83FA]
8009	FA			
800A	83			
800B	3E		MVI A, 73	Семисегментный код буквы
800C	73			P ⟶ A
800D	32		STA 83FB	A ⟶ [83FB]
800E	FB			
800F	83			
8010	3E		MVI A, 7D	Семисегментный код цифры
8011	7D			6 ⟶ A
8012	32		STA 83FC	A ⟶ [83FC]
8013	FC			
8014	83			
8015	3E		MVI A,3F	Семисегментный код O ⟶ A
8016	3F			
8017	32		STA 83FD	A ⟶ [83FD]
8018	FD			
8019	83			
801A	AF		XRA A	O ⟶ A
801B	32		STA 83FE	A ⟶ [83FE], гашение индикатора
801C	FE			
801D	83			
801E	32		STA 83FF	A ⟶ [83FF], гашение индикатора
801F	FF			
8020	83			
8021	76		HLT	Останов

Урок 15

НЕКОТОРЫЕ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ И АЛГОРИТМЫ

Устройства, основанные на МП, широко используются для выполнения математических операций в различных приборах /например, в электронных калькуляторах/. Однако программы к микропроцессорным системам, используемые в данном курсе, ограничены применением положительных целых чисел от 0 до 255 и обеспечивают лишь простые логические действия, сложение и вычитание. Этот урок рассказывает о методах, которые позволяют представить более широкий диапазон чисел и выполнять более сложные математические операции.

ДВОИЧНОЕ ПРЕДСТАВЛЕНИЕ

Рассмотрим представление положительных и отрицательных чисел, используя 8-битное слово. Поскольку невозможно представить больше чем 256 различных чисел с помощью 8 бит, то диапазон действия ограничен от -127 до +127. Первые 128 чисел /от 0 до 127/ /7F в шестнадцатеричном исчислении/ определяются как положительные. Отрицательные же числа получаются при отсчете от 0 в обратную сторону. В случае такого реверсивного счетчика, если регистр равен 0000 0000 и счет уменьшается, следующее число будет 1111 1111 /FF/. Поэтому FF представляется как -1. Это представление называется дополнением до двух /двоичное дополнение/. Представим двоичное дополнение от -8 до +7:

7	0000	0111
6	0000	0110
5	0000	0101
4	0000	0100
3	0000	0011
2	0000	0010
1	0000	0001
0	0000	0000
-1	1111	1111
-2	1111	1110

-3	1111	1101
-4	1111	1100
-5	1111	1011
-6	1111	1010
-7	1111	1001
-8	1111	1000

Заметим, что старшей значащей цифрой определяется знак.Если это 0, то число положительное, если 1 - отрицательное. Процедура вычисления с дополнением до двух предельно проста. Для положительных чисел двоичное дополнение и двоичное представление совпадают /как показано первыми восемью строчками/. Для отрицательных чисел процедура вычисления двоичного дополнения следующая:

1. Записать двоичное представление абсолютного значения, т.е. для -5 записать 0000 0101.

2. Выполнить дополнение полученного числа, т.е. 0000 0101=1111 1010 /это называется дополнением до единицы/.

3. Добавить единицу для получения двоичного дополнения /1111 1010=1111 1011=-5/.

Для примера рассмотрим двоичное дополнение числа 1111 1011:

$$\overline{1111\ 1011} = 0000\ 0100$$
$$0000\ 0100+1=5,$$

т.е. 1111 1011 является представлением -5.

Заметим, что число 1111 1011 может также интерпретироваться как 251_{10}, если бы это рассматривалось в двоичной системе, а не в системе с дополнением до двух. Поэтому необходимо постоянно помнить,в какой системе ведутся вычисления.

Система двоичного дополнения оказывается очень удобной для арифметики. Двоичнодополненные числа при сложении, вычитании, умножении или делении выдают результат в двоичном дополнении. Это свойство используется в процессорных системах при представлении положительных и отрицательных чисел.

Хотя двоичное дополнение обеспечивает представление отрицательных чисел, радиус действия этого метода ограничивается целыми числами, по модулю не превышающими 129. Этот радиус может быть увеличен несколькими путями в зависимости от того, какая длина слова и какая точность необходимы.

ДВОЙНАЯ ТОЧНОСТЬ

Простейшим способом увеличения радиуса чисел может быть простое увеличение числа бит, используемых для представления каждого числа. Часто это осуществляется с помощью использования пары слов для того, чтобы представить одно число /рис.1/.

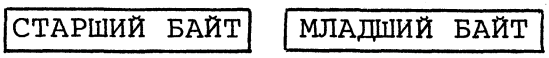


Рис.1. Двойная точность

В процессоре КР580ИК80 это может быть осуществлено в помощью инструкций, использующих пару регистров за один раз, которые могут работать одновременно с 16 битами. Использование двух слов для одного числа называется двойной точностью. В 8-битном процессоре это расширяет радиус до 65535 или $\pm$32767.

МЕТОД ФИКСИРОВАННОЙ ЗАПЯТОЙ

Двойная точность расширяет радиус величин, но для чисел меньше 1 вопрос остается открытым.

На рис.2 показано представление с фиксированной запятой. В этом примере 2 байта используются для запоминания числа. Первый байт определяется как левый байт от двоичной запятой /действительно

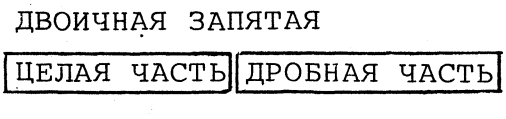


Рис.2. Представление числа с фиксированной запятой

двоичной запятой/, второй байт - справа от запятой, что позволяет представить и такие малые числа, как $2^{-8}=1/256$, и дробные числа, такие как 3,17. Решение, однако, ограничивается 1/256 /т.е. 0,004/, при этом радиус находится в пределах $\pm$127.

МЕТОД ПЛАВАЮЩЕЙ ЗАПЯТОЙ

При довольно большом объеме памяти, отведенном под представление чисел /16 бит/, методом фиксированной запятой невозможно представить такие числа, как 360 000 000 000 или 0,000000297. Эти числа могут быть легко представлены при использовании мантиссы и порядка, т.е. методом плавающей запятой. Число 360 000 000 000, например, может быть записано как $0{,}36 \cdot 10^{12}$ /мантисса будет 0,36 и порядок 12/, а 000000297 может быть записано как $0297 \cdot 10^{-6}$ /0297 - мантисса, 6 - порядок/.

Будем считать, что 2 байта используются для представления числа так, как показано на рис.3. Один байт есть мантисса, другой - порядок. Радиус значений может быть определен, полагая, что как мантисса, так и порядок запоминаются в форме двоичного дополнения, т.е. он равен $\pm 10^{\pm 127}$.

Рис.3. Представление числа с плавающей запятой

Метод плавающей запятой используется для представления широкого диапазона чисел. Числа объемом более 2 байт используются для того, чтобы получать большее разрешение /больше чисел в мантиссе/.

Заметим, что, как и во всех представлениях, тип представления должен быть определен для того, чтобы распознать число. Подобные 2 байта данных отличаются один от другого, если представляются как пара в системе двоичного дополнения методами фиксированной или плавающей запятой. Тип представления должен быть заложен в программе.

ДЕСЯТИЧНОЕ ПРЕДСТАВЛЕНИЕ

Большинство микропроцессорных систем имеют десятичные входные-выходные устройства, такие как клавиатура и дисплей /микролаборатория является исключением, так как там используется шестнадцатеричное представление/. Так как десятичная система более привычна для большинства людей, процессор должен быть приспособлен и к ней.

Затруднения вызывает представление десятичных чисел в двоичной системе процессора. Для примера допустим, что десятичное число 28 считывается с клавиатуры. Число может быть преобразовано в двоичный эквивалент 0001 1100 /1С/, однако, если это число показывается на дисплее /десятичном/, то оно должно быть вновь преобразовано к десятичной системе, к цифрам 2 и 8.

Альтернативный метод - перевод каждой десятичной цифры в независимое 4-битное слово /символ/. Два 4-битных слова затем объединяются в один байт. Таким образом, 28 будет кодироваться как 0010 1000. Этот код называется двоично-десятичным кодом /ДДК/. Заметим, что все двоичные символы с 1010 и 1111 никогда не используются в двоично-десятичном представлении. ДДК часто используется в системах, которые имеют десятичные порты ввода/вывода, так как это освобождает от процесса десятично-двоичного преобразования. Одно неудобство метода - неэффективное использование объема памяти. Самое большое десятичное число, которое может быть запомнено в байте, используя ДДК,- 99, тогда как в чисто двоичном виде максимальное число равно 255. Арифметика также плохо подходит для ДДК, так как это "неестественная" система чисел. Однако большинство МП обеспечиваются специальными инструкциями для связи с ДДК.

АЛФАВИТНОЕ ПРЕДСТАВЛЕНИЕ

Многие процессорные системы должны работать не только с числами, но и с буквами. Компьютерный терминал, например, должен читать символы с клавиатуры и посылать их на компьютер. Буквы должны быть как-то представлены двоичными символами.

Наиболее часто применяющийся для этого код, соответствующий ГОСТ 13052-67, приведен в табл.1. Каждому символу ставится в соответствие двоичное значение. Этот код используется, в основном, для обес-

Таблица 1

Код	Символ	Код	Символ	Код	Символ	Код	Символ
00	ПУС	20	пробел	40	G	60	Ю
01	НЗ	21	!	41	A	61	А
02	НТ	22	,,	42	B	62	Б
03	КТ	23	#	43	C	63	Ц
04	КП	24	¤	44	D	64	Д
05	КТМ	25	%	45	E	65	Е
06	ДА АСК	26	&	46	F	66	Ф
07	ЗВ	27	,	47	G	67	Г
08	ВШ	28	(	48	H	68	Х
09	ГТ	29	)	49	I	69	И
OA	ПС	2A	*	4A	J	6A	Й
OB	ВТ	2B	+	4B	K	6B	К
OC	ПФ	2C	,	4C	L	6C	Л
OD	ВК	2D	-	4D	M	6D	М
OE	РУС	2E	,	4E	N	6E	Н
OF	ЛАТ	2F	/	4F	O	6F	О
10	АР1	30	0	50	P	70	П
11	(СУ1)	31	1	51	Q	71	Я
12	(СУ2)	32	2	52	R	72	Р
13	(СУ3)	33	3	53	S	73	С
14	СТП (СУ4)	34	4	54	T	74	Т
15	НЕТ	35	5	55	U	75	У
16	СИН	36	6	56	V	76	Ж
17	КБ	37	7	57	W	77	В
18	АН	38	8	58	X	78	Ь
19	КН	39	9	59	Y	79	Ы
1A	ЗМ	3A	:	5A	Z	7A	З
1B	АР2	3B	;	5B	[	7B	Ш
1C	РН4	3C	<	5C	V	7C	Э
1D	РН3	3D	=	5D	]	7D	Щ
1E	РН2	3E	>	5E	^	7E	Ч
1F	РН1	3F	?	5F	-	7F	ЗБ

печения специальных функций. Код "OA", например, используется для вызова линии загрузки на дисплей или печатающее устройство.

Общая проблема программирования - перевод данных из одного кода в другой. Для примера рассмотрим проблему записи шестнадцатеричных цифр на дисплей /семисегментный/. Для того чтобы определить, какие сегменты будут включаться на дисплее, для каждого символа требуется преобразование из двоичного кода в семисегментный. Это осуществляется с помощью метода, называемого table look up. Сегментные коды для каждого символа запоминаются как список в памяти, представленный таблицей. Чтобы перевести двоичный код в соответствующий семисегментный, достаточно взглянуть на таблицу "looked up" /рис.4/.

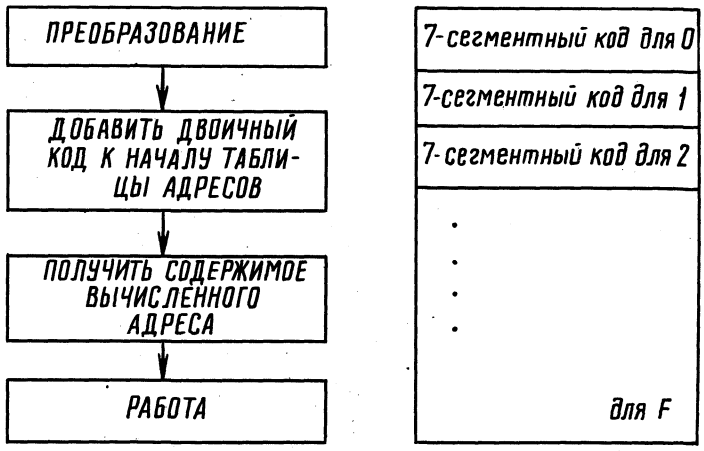


Рис.4. Семисегментное преобразование

Первый выход в табл.1 содержит семисегментный код для 0, следующий выход - для 1, и так далее. Сначала двоичное число преобразуется и добавляется к адресу первого выхода в табл.1. Результатом является адрес таблицы выходов, содержащих желаемый семисегментный код. Содержание адресуемой ячейки затем читается, и на этом преобразование закончено.

МАТЕМАТИЧЕСКИЕ АЛГОРИТМЫ

Калькуляторы, выполненные на основе МП, осуществляют деление, умножение, определение синуса, косинуса и много других математических операций. Чисто арифметические же возможности процессора КР580ИК80 /и большинства других 8-битных процессоров/ сводятся только к сложению и вычитанию.

Однако,используя только простые операции сложения и вычитания, можно представить любые сложные математические функции. Умножение может быть представлено очень просто с помощью серии сложений и сдвигов.Для того чтобы посмотреть, как все это делается, рассмотрим процесс умножения "вручную". Множимое умножается только на одну цифру множителя за один раз. Результат с каждой последующей цифры множителя сдвигается влево на одну позицию и результаты складываются. Подобная процедура осуществляется и для двоичных чисел. Число, умноженное на 1, равно самому себе, а число, умноженное на 0, равно 0:

```
    1011
x   1001
    1011
   0000
  0000
 1011
1100011
```

Таким образом, операцию умножения можно осуществить только с помощью сдвига и сложения , так же выполняется и деление: используя сдвиг и вычитание.

Вычисление таких функций, как sin x или cos x, является более сложной операцией. Алгоритм представления функции синуса такой:

$$\sin x = x - \frac{x^3}{3\cdot 2} + \frac{x^5}{5\cdot 4\cdot 3\cdot 2} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$$

На практике может быть вычислено только конечное число слагаемых. Это приводит к аппроксимации результата, но очень точной, если вычисляется большое число термов /что не проблема для МП/.

Последовательность, описанная выше, обеспечивает алгоритм для вычисления синуса угла. Для каждой математической функции также существует алгоритм, который можно вычислить, используя только элементарные операции МП.

ДИАГНОСТИКА И ПОИСК НЕИСПРАВНОСТЕЙ

Урок 16

ПОРТАТИВНЫЕ ДИАГНОСТИЧЕСКИЕ УСТРОЙСТВА. ПОИСК НЕИСПРАВНОСТЕЙ В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ

По оценкам специалистов, в 1979 г. во всем мире в различных приборах использовалось 10 блн. цифровых ИС. Средняя частота отказов ИС за год составляет 1%, среднее количество отказов за год равно 10 млн. А если учесть, что средняя стоимость одного ремонта около 100 рублей, то несложно подсчитать, что каждый год на ремонт тратится 1 блн.руб. Таким образом, получается, что на ремонт прибора тратится ежегодно сумма большая, чем стоимость всех микросхем, входящих в его состав.

Приборы, построенные на базе МП, имеют большую надежность и лучшие характеристики. Но при этом появляются новые проблемы обслуживания, диагностики и ремонта, необходимость новых диагностических приборов, а также методов проверки. Такими диагностическими приборами могут быть логический пробник и логический стимулятор, сигнатурный анализатор, анализатор логических состояний и осциллограф.

Логические пробники и логические стимуляторы предназначены для определения и стимулирования уровней и импульсов в логических цепях. Эти при-

боры очень эффективны при диагностике цифровых устройств, удобны в эксплуатации.

ЛОГИЧЕСКИЕ ПРОБНИКИ

Логические пробники определяют состояние логической цепи. С помощью простых ламповых индикаторов они показывают логическое состояние цепи и позволяют определить короткие импульсы. Проведем эксперимент, в котором логические сигналы цепей микролаборатории используются для демонстрации применения логического пробника КИТ-1.

1. Убедитесь, что микролаборатория включена и светится дисплей. Если дисплей не светится, проверьте выключатель сети и нажмите кнопку СБРОС.

2. Подключите проводники питания логического пробника к источнику питания +5 В. Источник питания проверяемого устройства может использоваться в том случае, если он обеспечивает дополнительный ток /60 мА/ пробника.

3. Лампочка на конце пробника выдает информацию о логическом состоянии цепи. Каждая точка цепи называется узлом. Все точки, непосредственно соединенные проводом между собой, являются частями одного узла. Когда металлический кончик пробника ничего не касается, лампочка горит тускло. Это соответствует высокоимпедансному состоянию или обрыву цепи. Если такая индикация появляется, когда кончик пробника находится в контакте с логическим узлом, это означает наличие "серого" уровня,т.е. уровня, значение которого находится между лог.0 и лог.1 /рис.1/. Если бы

2,4 В	СВЕЧЕНИЕ ЯРКОЕ	ЛОГ.1
0,8 В	СЛАБОЕ СВЕЧЕНИЕ	"СЕРЫЙ" УРОВЕНЬ
0	ЛАМПОЧКА НЕ ГОРИТ	ЛОГ.0

Рис.1. Пороги ТТЛ для логического пробника КИТ-1

этот узел был шиной с тремя состояниями /со всеми запрещенными выходами устройство/, то наличие "серого" уровня было бы нормальным явлением. Но наличие "серого" уровня на обычном логическом узле указывает на неисправность.

4. Прикоснитесь щупом пробника к шине "земля". Лампочка на пробнике должна погаснуть, индицируя лог.0.

5. Прикоснитесь щупом пробника к шине +5 В. Лампочка загорается ярче, индицируя лог.1.

6. Используя логический пробник, прозондируйте выходы 13, 12, 11 микросхемы D37. Мерцающий свет указывает на наличие быстроизменяющихся сигналов в этих цепях. Если одна из этих шин сканирования клавиатуры неисправна, то с помощью пробника можно легко выявить это.

7. Теперь подключите пробник к выходу 2 микросхемы D55. Лампочка на пробнике должна ярко загореться, индицируя наличие лог.1.

8. Теперь нажмите кнопку F клавиатуры. Мерцающий свет лампочки пробника, присоединенной к ножке 2 микросхемы D55, указывает, что кнопка F периодически опрашивается и МП "знает" ее состояние.

С помощью логического пробника возможно зондирование узлов микролаборатории. Используя схему микролаборатории, прозондируйте другие узлы микро-ЭВМ. Сравните результаты с ожидаемыми, основанными на знаниях о микролаборатории и программе монитора. Проверьте линии шин, входные и выходные порты, управляющие цепи. Вы обнаружите, что все эти схемы работают.

Итак, логический пробник - это самостоятельное, легкое в применении устройство, предназначенное для проверки логических узлов. Пробник может использоваться для проверки цельности цепей, дешифрации адреса, синхронизации работы переключателей и шин. Он также используется для проверки операционных характеристик лаборатории в шаговом режиме.

ЛОГИЧЕСКИЕ СТИМУЛЯТОРЫ

Логический стимулятор из набора КИТ-1 является внутрисхемным стимулирующим устройством. Он автоматически выдает импульсы требуемой логической полярности и амплитуды.

Проведем эксперимент с использованием логического стимулятора для проверки узлов микролаборатории.

1. Подсоедините логический стимулятор к клеммам питания микролаборатории.

2. Нажмите и отпустите кнопку на стимуляторе. Наблюдайте с помощью логического пробника, подключенного к стимулятору, одиночную вспышку на индикаторе.

3. Нажмите и задержите в нажатом положении кнопку на стимуляторе. Удостоверьтесь, что индикатор на логическом пробнике начинает быстро вспыхивать. Теперь стимулятор "запрограммирован" на выдачу последовательности импульсов.

4. Прикоснитесь кончиком стимулятора к ножке 9 микросхемы D 53. Подавайте последовательность импульсов. В динамике услышите потрескивание /рис.2/.

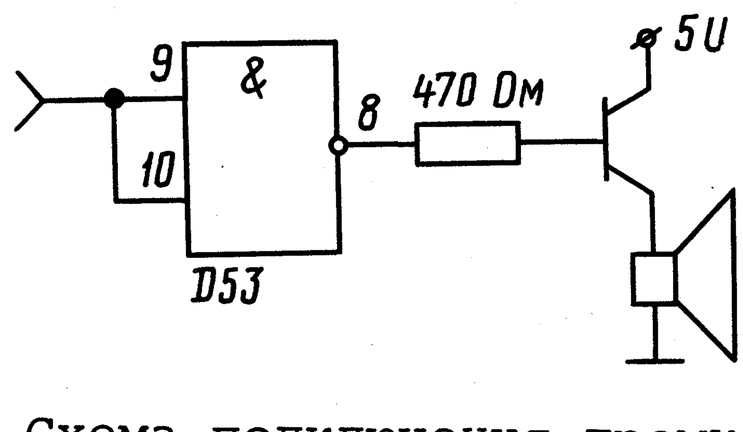


Рис.2. Схема подключения громкоговорителя в микролаборатории

5. Попробуйте стимулировать шины "питание" и "земля". С помощью логического пробника будете наблюдать, что стимулятор не может их перевозбудить. Это происходит потому, то источник питания имеет гораздо меньший выходной импеданс, чем логические устройства, для возбуждения которых предназначен стимулятор.

Итак, логический стимулятор может подавать на схему либо одиночные импульсы, либо пачки импульсов. Стимулятор может также использоваться как генератор различных разрешающих сигналов и сигналов синхронизации.

Урок 17

СИГНАТУРНЫЕ АНАЛИЗАТОРЫ

Сигнатурные анализаторы - это специализированные приборы, используемые при разработке, производстве и обслуживании радиоизмерительной аппаратуры. Сигнатурный анализ, производимый с помощью сигнатурного анализатора, позволяет с высокой степенью достоверности локализовать дефектные логические узлы.Сигнатурный анализатор преобразует длинные и сложные потоки последовательных двоичных данных в логических узлах микропроцессорных систем в 4-значные сигнатуры. Сравнивая их с эталонными сигнатурами, можно судить о правильности работы узла. Для сигнатурного анализа используется набор символов, содержащий 16 знаков /0-9, A, C, F, H, P, U/. Все знаки можно легко отличить друг от друга. Это позволяет исключить ошибку оператора, которая может возникнуть из-за сходства в семисегментном изображении некоторых шестнадцатеричных знаков /например 8 и B/.

В общем случае, контрольные точки в предполагаемой зоне неисправности зондируются сигнатурным анализатором до обнаружения сигнатуры, отличной от эталонной, записанной в инструкции по обслуживанию. Затем цепь прохождения сигнала прослеживается в обратном порядке до обнаружения правильной сигнатуры, что облегчает локализацию дефекта. После обнаружения дефектного узла уточняется дефектный компонент. Сигнатуры, соответствующие цифровым узлам, аналогичны информации о напряжении и форме импульса для аналоговой схемы. Сигнатуры в узлах получаются благодаря тестовой программе, которая генерирует соответствующие испытательные стимулирующие сигналы и обеспечивает проверку отдельных частей схем.

Определение эталонных сигнатур заключается в исследовании заведомо исправного прибора и записи сигнатур каждого узла. Такой подход используется, поскольку очень трудно предусмотреть то огромное количество сложной информации, которое содержится в каждой сигнатуре. Снятие полного комплекта эталонных сигнатур занимает много времени, и при этом могут быть допущены различные ошибки. Если при коррекции схемы прибора изменится содержимое хотя бы одного слова ПЗУ, то могут измениться все или большин-

ство сигнатур. Поэтому всегда необходимо удостовериться в том, что номер прибора соответствует используемой документации. Такого понятия, как "почти правильная сигнатура" не существует; она может быть либо правильной, либо неправильной.

Для получения сигнатур на сигнатурный анализатор должны быть поданы следующие сигналы: ПУСК, СТОП, ТАКТ, ДАННЫЕ. На вход ДАННЫЕ подается сигнал с проверяемого узла. Сигналы ПУСК и СТОП, формируемые проверяемым устройством, определяют соответственно начало и конец сигнатурной обработки поступивших сигналов. В этом "окне измерения" данные записываются в регистр анализатора синхронно с сигналом ТАКТ, также снимаемым с проверяемого устройства. В документации по обслуживанию проверяемого прибора указываются точки съема этих сигналов.

В экспериментах, описанных ниже, проверяются сигнатуры отдельных узлов микролаборатории в нескольких режимах и рассматривается связь сигнатур с частями схемы.

СВОБОДНЫЙ РЕЖИМ МИКРОПРОЦЕССОРНОЙ ЛАБОРАТОРИИ

При свободном режиме или режиме СВОБОДНЫЙ ПРОГОН минимальная часть схемы /в этом случае - МП/ проверяет по возможности большую оставшуюся часть схемы. Эту часть схемы часто называют ее ядром. В этом случае необязательно, чтобы ПЗУ, ОЗУ и устройства ввода/вывода были работоспособными. Любая ветвь обратной связи /такая, как шина данных или прерывания/, идущая к МП, должна разрываться так, чтобы на его работу не влияли неправильные или неожиданные сигналы.

Этот метод позволяет получить испытательные стимулы, необходимые для измерения сигнатур. Преимуществом использования такого режима является то, что для него требуется исправность только очень небольшой части схемы /МП, генератор тактовых импульсов и схема питания/. Опишем получение сигнатур в свободном режиме.

Смените перемычку 1 на панели шины данных на перемычку 2. При этом разрывается цепь шины данных между МП и остальной частью схемы, и программа из памяти не будет считываться в МП. Таким образом, обеспечивается возможность свободного режима. На шине данных всегда будет команда MOVA, A /11111110/, которая является "пустой" командой. МП будет считывать эту команду, а затем приращивать адрес для считывания следующей команды. Вследствие того, что шина данных системы отключена от МП, он каждый раз исполняет эту команду/при каждом адресе/.Поэтому на адресной шине МП происходит непрерывный перебор всего поля адресов /2^{16}/.

В микролаборатории свободный режим может не выполняться из-за неисправности небольшого числа схем. К ним относятся источник питания, испытательные перемычки, генератор тактовых импульсов и схемы, подсоединенные непосредственно к управляющим и адресным выводам МП. В качестве первой /и основной/ границы деления схемы при локализации неисправности в микропроцессорной системе можно выбрать ядро схемы. Если система не работает в свободном режиме, то проблема поиска неисправности ограничена небольшим числом компонентов.

Получение сигнатур:

а/ подсоедините зажимы "⊥" сигнатурного анализатора к разъему заземления микролаборатории /↓/, зажимы ПУСК и СТОП к выводу A15 /D19/4/, а зажим ТАКТ - к выводу DBIN /D2/9/;

б/ установите переключатели ПУСК, СТОП в сигнатурном анализаторе в положение Z, а переключатель ТАКТ - в положение ⎍. Убедитесь, что индикатор СТРОБ в анализаторе вспыхивает, индицируя формирование "окна измерения", которое имеет размер полного адресного поля /64 К бита/.

Сигнал ДАННЫЕ вводится в сигнатурный анализатор при каждом положительном перепаде /фронте/ сигнала, т.е. 65536 раз в течение каждого окна измерения /рис.1/;

в/ сигнатурный анализатор теперь может быть использован для проверки правильности перебора адресов в свободном режиме. Подсоедините пробник данных анализатора 817, который кроме того работает

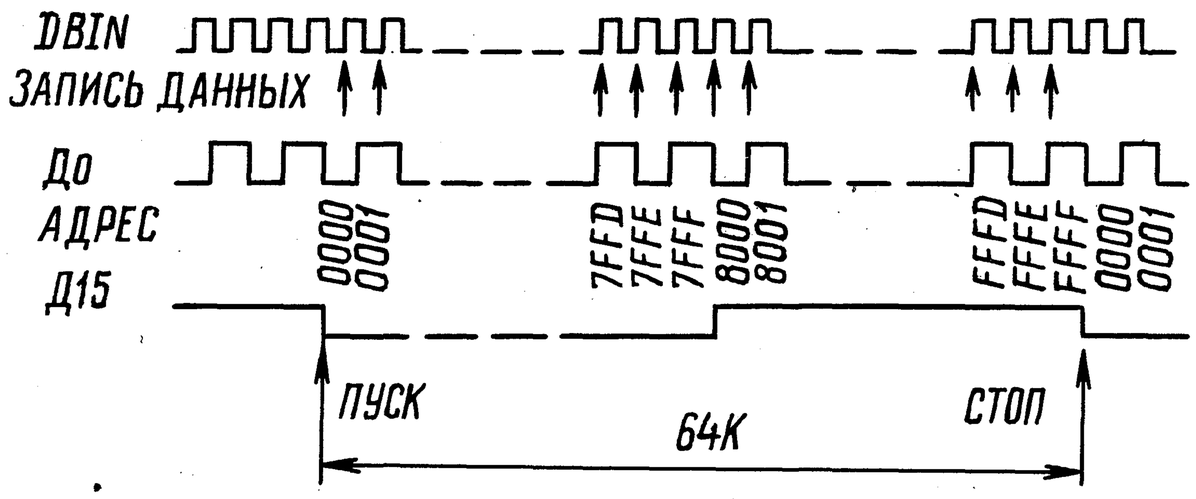


Рис.1. Запись данных в анализатор при проверке адресного поля в свободном режиме работы

как логический пробник, к земле. Убедитесь, что индицируется сигнатура 0000, а индикатор данных в пробнике не светится. Сигнатура нулевого уровня всегда будет 0000;

г/ для получения сигнатуры на шине питания /V_n / подключите пробник данных к линии +5 В. Убедитесь, что индицируется сигнатура 0001 - характеристическая сигнатура логической единицы при заданных ранее положениях переключателей сигнатурного анализатора и для фиксированного режима работы микролаборатории. Это можно видеть, изменяя положения переключателей активного фронта ПУСК, СТОП сигнатурного анализатора и наблюдая, что сигнатура при этом изменяется. Таким образом, сигнатура на шине 5 В -0001 - указывает на то, что такая проверка является верной и что "ядро" схемы скорее всего работает правильно в свободном режиме. Если сигнатура получается отличной от 0001, то можно предполагать или неправильные положения переключателей в анализаторе, или дефект в ядре схемы или в перемычках ядра, обеспечивающих свободный режим. Такая проверка позволяет сократить число схем, которые могут считаться потенциально неисправными;

д/ проверьте сигнатуры на линиях адресной шины A0-A15, удостоверьтесь, что оси соответствуют записям в таблице. Это означает, что МП D11, адресные буферы D16-D19 работают правильно;

е/ получите сигнатуры на выводах адресного дешифратора /микросхема D27, выводы 7,6,5,4,12,3,13, 1, 15 и микросхемы D29-D31 выводы 8/, сравните их с записями в табл.1. Эти сигнатуры показывают, что адресный дешифратор и микросхемы D23-D25, D29-D31 работают правильно;

ж/ все сигнатуры в табл.1 сообщают пользователю что-то полезное. Отсутствие при проверке у нескольких устройств сигнатуры обусловлено тем, что эти устройства не могут быть проверены в режиме свободной работы: либо отсутствует активность на их выходах /для этого им необходим сигнал записи/, либо данные на них непредсказуемы.

Таблица 1

Контакт	Микросхемы								Контакт	Разъем
	16	17	18	19	27	29	30	31		
1					755P	HPP1	HPP1	HPPO	49	0000
2						HPP1	1292	1292	44	5555
3					7AOC	1292	HAP6	HAP6	46	CCCC
4	52F8	7F7F	HPPO	755U	0001B	HAP6	3C97	3C97	47	7F7F
5					0001B	755U	755P	3826	48	5H21
6					0001B	755U	755P	3826	45	OAFA
7	OPFH	CCCC	1293	3827	0001B				43	OPFH
8						FC75	51FP	6C23	52	52F8
9	OAFA	5555	HAP7	2H70					51	HC89
10									53	2H70
11						3C97	3826	755P	25	HPPO
12	5H21	UUUU	3C96	HC89	0001B	3826	3826	755P	27	755U
13					AF8U				29	HAP7
14									23	3C96
15					755P				42	3827
16									50	1293

Сигнатуры на таких выводах не могут быть полезными для диагностирования.Для иллюстрации проделайте операции, приведенные ниже.

Проверьте сигнатуру любой из линий шины данных. Эта сигнатура - результат действия всех устройств микролаборатории, выходы которых включены на шину данных в течение перебора адресного поля размером 64 К бита. Выключите питание микролаборатории, а затем снова включите его. При этом сигнатура обычно изменяется из-за того, что в ОЗУ /в результате цикла отключения питания/ теперь записались другие /произвольные/ данные. Попробуйте повторить эту операцию еще раз. Сигнатура, вероятно, снова изменится.

В свободном режиме МП считывает каждый возможный адрес из 64 К адресов. Если взять "окно измерения", соответствующее адресам ПЗУ /т.е. моментам времени, когда на шине данных присутствует информация только с ПЗУ/ - адреса 0000 07FF, то и сигнатура будет определяться только содержимым ПЗУ. Возможен другой способ организации проверки ПЗУ или другого устройства микролаборатории - программный.

Смените перемычку 2 в панели на перемычку 1 /тем самым мы переходим в обычный режим работы микролаборатории/. Подключите входы ПУСК и СТОП сигнатурного анализатора к выводу 20 микросхемы D37. Установите вход ПУСК в положение Z, а СТОП - в положение S.

Таблица 2

Адрес	Содержимое	Мнемоника	Комментарии
8000	3E	MVI A, 80	Программирование порта ввода/вывода
8001	80		
8002	D3	OUT F, B	
8003	FB		
8004	21	LXI H, 0000	Занесение начального адреса опрашиваемого ПЗУ
8005	00		
8006	00		
8007	11	LXI D, 0002	Загрузка счетчика
8008	00		
8009	02		
800A	7E	MOV A, M	Формирование импульсов до открытия "окна"
800B	7E	MOV A, M	
800C	7E	MOV A, M	
800D	3E	MVI A, 00	Открытие "окна"
800E	00		
800F	D3	OUT F9	
8010	F9		
8011	7E	MOV A, M	Обращение к ячейкам ПЗУ
8012	23	INX H	
8013	1D	DCR E	
8014	C2	JNZ 8011	
8015	11		
8016	80		
8017	15	DCR D	
8018	C2	JNZ 8011	
8019	11		
801A	80		
801B	3E	MVI A	Закрытие "окна"
801C	FE		
801D	D3	OUT F9	
801E	F9		
801F	C3	JMP 8004	Переход на начало
8020	04		
8021	80		

Подключите вход ТАКТ анализатора к выводу 10 микросхемы D2 и установите переключатель ТАКТ на передней панели анализатора в положение Z.

Введите программу, приведенную в табл.2.

Переведя эту программу в мнемонические коды, можно обнаружить, что она обеспечивает формирование "окна измерения" на выводе 20 микросхемы D37 /это достигается путем вывода соответствующих данных в порт - D37/.

Кроме того, программа обеспечивает перебор всех адресов первого ПЗУ.

Подсоедините пробник данных к шине +5 В и убедитесь в наличии правильной сигнатуры /4596/.

Подсоединяя пробник данных к каждому из выводов микросхемы D45, получите сигнатуры, соответствующие данному ПЗУ /рис.2/.

Аналогичным способом можно проверить микросхемы D46, D47, а также другие схемы микролаборатории. В результате этого теста можно получить большое количество информации.

Вы проверили ПЗУ и можете гарантировать, что оно хорошо работает.

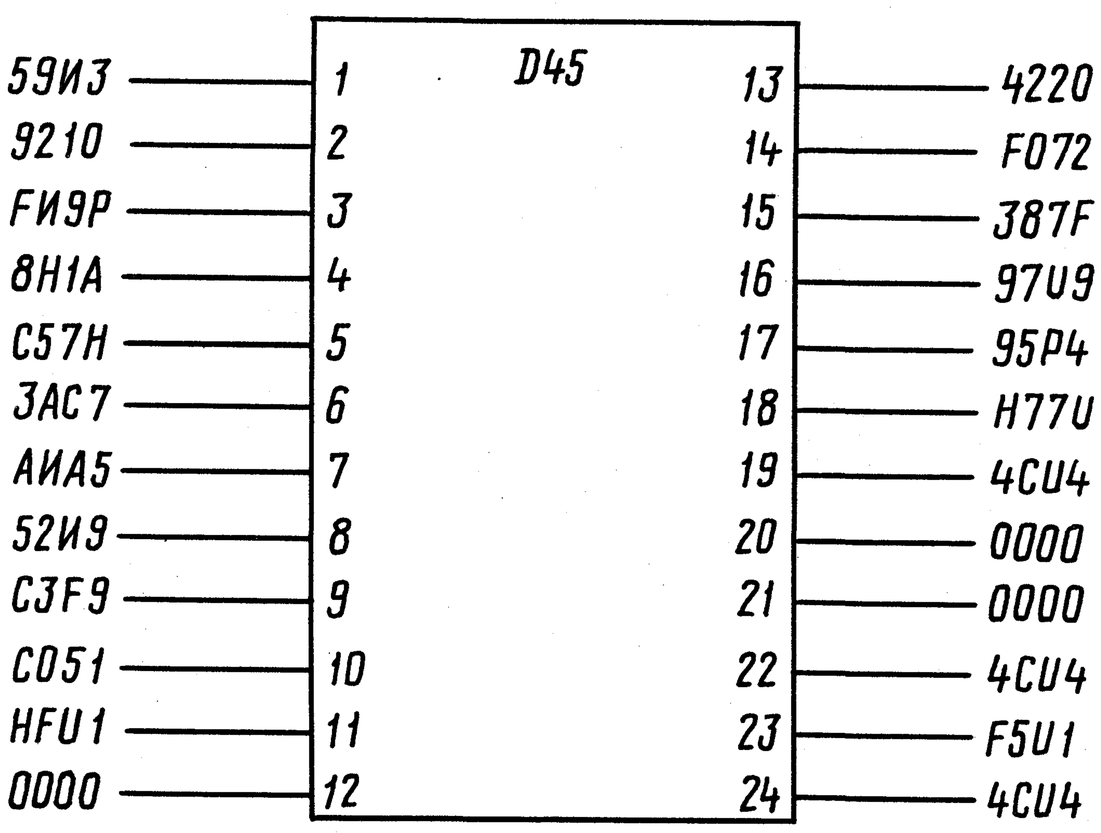


Рис.2. Сигнатуры, соответствующие выводам микросхемы D45

При адресации ПЗУ не было никаких перекрытий шин /т.е. два или более устройств не работали на одной шине/, а это говорит об отсутствии серьезных дефектов шины данных.

Урок 18

ЛОГИЧЕСКИЕ АНАЛИЗАТОРЫ

При разработке и отладке аппаратных и программных средств микропроцессорных устройств применяются логические анализаторы, показывающие логическое состояние в зависимости от числа прошедших тактовых импульсов и позволяющие проверять работу микропроцессорных устройств в реальном времени.

Логический анализатор - прибор, осуществляющий сбор логических сигналов в реальном времени по отношению к событию запуска и представляющий эти сигналы в различных формах.

Событие запуска - это событие или несколько событий, управляющих процессом сбора логических сигналов. Событие запуска может быть выработано при опознавании слова.

Под опознаванием понимается совпадение заранее введенного оператором слова со словами анализируемого потока данных.

Из выпускаемых в настоящее время логических анализаторов наиболее удобен для отладки микропроцессорных устройств анализатора потока цифровых данных.

Прибор имеет 32 входных канала, что позволяет анализировать данные на информационных магистралях микро-ЭВМ. Анализатор потока цифровых данных дает возможность отлаживать сложные программы, управляющие работой микро-ЭВМ.

Программы по структуре подразделяются на линейные, разветвляющиеся, циклические и подпрограммы /рис.1/.

Процесс выполнения программы выражается в смене состояний на шинах контролируемого устройства. Для отладки программных и аппаратных средств важно знать эти состояния при выполнении определенного участка программы. Поскольку оператор не может наблюдать смену состояний во время выполнения программы, необходимо запоминающее устройство /память сбора данных/ для записи этой информации с целью последующей индикации. Память сбора данных имеет ограниченный объем /32x64 бита/, поэтому анализиро-

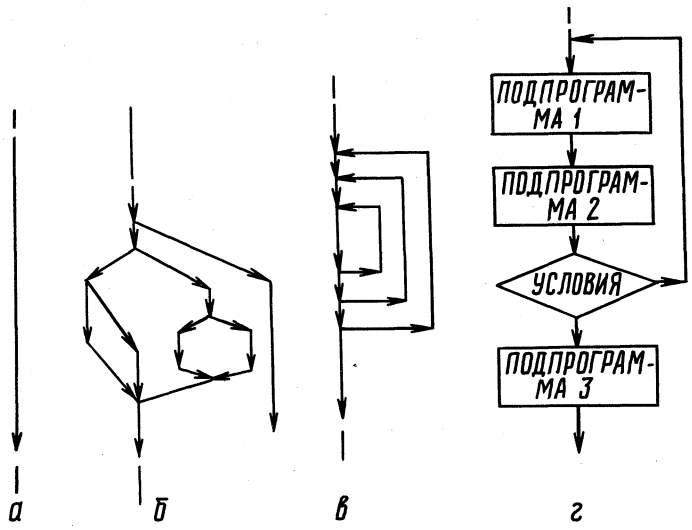


Рис.1. Основные структуры программ: а - линейная; б - разветвляющаяся; в - циклическая; г - подпрограммы

вать программу, которую выполняет проверяемое устройство, приходится по частям. Необходимо четко определить момент начала сбора информации. Для определения этого момента следует пользоваться условиями просмотра. Под условиями просмотра понимаются выделенные цифровые коды состояния контролируемых шин, запоминаемые в анализаторе и используемые для распознавания момента возникновения этих состояний на шинах.

Среди условий просмотра особое место занимают точки траектории. Точки траектории определяют ветвь программы, на которой находится анализируемый участок.

При отладке программы с линейной структурой /см.рис.1, а/ в большинстве случаев достаточно задать всего одну точку траектории, через которую должен пройти линейный поток, и дать анализатору сигнал о начале сбора информации. Эта точка и будет событием запуска.

При отладке программ с разветвляющейся структурой /см.рис.1, б/ необходимо задавать уже несколько точек траектории и порядок их распознавания для выделения заданной области процесса выполнения программы. Последняя из этих точек будет событием запуска.

При отладке программ с циклической структурой /см.рис.1, в/ кроме нескольких точек траектории и их порядка целесообразно задавать количество повторений каждой точки траектории, чтобы выделить необходимое число повторений цикла и только после этого выработать сигнал для сбора информации.

При отладке программ с повторяющимися элементами /например, использование подпрограмм, макроопределений - см.рис.1, г/ целесообразно использование так называемого условия перезапуска. Условие перезапуска, так же как условия просмотра, представляет собой выделенный цифровой код. Встретив его в анализируемом потоке данных, анализатор игнорирует уже найденные и записанные в память сбора данных состояния с тем, чтобы снова начать поиск первой точки траектории /происходит перезапуск прибора/.

Анализатор потока цифровых данных обладает возможностью устанавливать событие запуска в начале /режим СТАРТ/, в конце /режим КОНЕЦ/ и в середине /режим ЦЕНТР/ анализируемой последовательности. В режиме ЦЕНТР анализатором запоминаются 31 состояние до и 32 состояния после события запуска, в режиме КОНЕЦ - 63 состояния до события запуска. Посредством выбора "неисправной" команды в качестве события запуска и последующего рассмотрения событий можно обнаружить неисправность.

Одной из оценочных характеристик при выполнении программы является фиксирование ее прохождения через какие-либо состояния. Такую возможность дает выборочный сбор кодов. Можно задавать несколько выборочных кодов, так же как и несколько точек траектории /как точки траектории, так и выборочные коды относятся к числу условий просмотра/, однако при распознавании выборочных кодов порядок их следования не имеет значения.

Смена состояний контролируемых шин характеризуется временем перехода из одного состояния в другое и числом промежуточных состояний, поэтому в анализаторе имеется возможность подсчета этих характеристик.

Для работы с анализатором потока цифровых данных необходимо хорошее знание программных и аппаратных средств проверяемых устройств.

Анализатор потока цифровых данных дает оператору информацию о

состоянии проверяемого устройства, что ускоряет процесс локализации неисправности.

Анализатор индицирует полученную информацию в различных системах счисления /двоичной, восьмеричной, шестнадцатеричной/ и в мнемоническом виде для МП КР580ИК80.

Для удобства работы оператора имеется возможность разбивки 32-разрядных слов на несколько полей, обозначаемых буквами A, B, C, D, E, F.

Запуск анализатора производится как положительным, так и отрицательным фронтом синхросигнала. Можно менять также и полярность логики /положительная и отрицательная/. Все эти параметры оператор задает в кадре ФОРМАТ.

Условия просмотра /в том числе точки траектории, выборочные коды, условие перезапуска/, а также условие подсчета времени или состояний задаются оператором в кадре УСЛОВИЯ ПРОСМОТРА. Содержимое памяти сбора данных в заданной системе счисления или в мнемоническом коде высвечивается в кадрах РЕЗУЛЬТАТ и МНЕМОНИКА соответственно. В кадре СРАВНЕНИЕ высвечиваются результаты сравнения информации, предварительно записанной в память прибора, с содержимым памяти сбора данных.После сбора информации на экране анализатора высветится часть программы-монитора.

Проведем эксперимент с использованием анализатора потока цифровых данных для проверки "Микролаб КР580ИК80":

1. Подключите к разъемам на передней панели анализатора пять 4-разрядных компараторов, начиная с правого /разряды 1...20/, а также компаратор тактовых импульсов.

2. Подключите зажимы компараторов /разряды 1..8/ к выводам данных микросхемы ПЗУ D45 на плате "Микролаб"/ /см.Приложение.Принципиальная схема/. Необходимо соблюдать соответствие разрядов ПЗУ и компараторов.

3. Подключите зажимы компараторов /разряды 9...17/ к выводам адресов микросхемы ПЗУ D45.

4. Подключите зажим компаратора тактовых импульсов к выводу 10 микросхемы D2.

5. Включите анализатор и "Микролаб".

6. Установите на экране анализатора кадр ФОРМАТ.

Распределите поля следующим образом:

XXXXXXXX XXXXXXXA AAAAAAAA BBBBBBBB

Установите для всех полей шестнадцатеричную систему счисления.

7. Установите на экране анализатора в кадре УСЛОВИЯ ПРОСМОТРА две точки траектории анализируемой программы следующим образом:

МЕТКИ	A	B	КОЛ-ВО
БАЗА	16	16	10
	043	XX	00001
СТАРТ	04C	XX	00010

8. Нажмите клавишу СБРОС "Микролаба" /не отпускайте ее/.

9. Нажмите и отпустите клавишу ПУСК анализатора. На экране анализатора появятся слова МЕД.ТАКТ.

10. Отпустите клавишу СБРОС "Микролаба". На экране анализатора появится изображение в соответствии с табл.1. В поле A высвечиваются адреса ячеек памяти, к которым обращается МП, в поле B - содержимое этих ячеек.

Таблица 1

Сбор данных в шестнадцатеричном коде

РЕЗУЛЬТАТ		
МЕТКИ	A	B
БАЗА	16	16
	043	21
СТАРТ	04C	C2
02	04D	49
03	04E	00
04	049	77
05	04A	23
06	04B	05
07	04C	C2
08	04D	49
09	04E	00
10	049	77
11	04A	23
12	04B	05
13	04C	C2
14	04D	49
15	04E	00
16	04F	21
17	050	C7
18	051	83
19	052	22

11. Нажмите клавишу ↑ из поля клавиш СДВИГ на передней панели анализатора. При этом на его экране происходит увеличение порядкового номера индицируемых слов /табл. 2/.

12. Отключите компараторы от анализатора, подключив на их место разъемы блока сопряжения.

Таблица 2

Просмотр данных

РЕЗУЛЬТАТ		
МЕТКИ	A	B
БАЗА	16	16
	043	21
19	052	22
20	053	E2
21	054	83
22	055	31
23	056	D1
24	057	83
25	058	CD
26	059	CO
27	05A	01
28	1C0	21
29	1C1	F4
30	1C2	83
31	1C3	11
32	1C4	F8
33	1C5	83
34	1C6	01
35	1C7	E9
36	1C8	01
37	1C9	7E

13. Подключите зажим, находящийся на конце плоского кабеля блока сопряжения, к МП на плате "Микролаба".

14. Нажмите клавишу СБРОС анализатора.

15. Установите в кадре УСЛОВИЯ ПРОСМОТРА значение условия [СТАРТ] следующим образом:

[СТАРТ] 0000000000000000
XXXXXXXX XXXXXXXX

16. Повторите операции п.8.

17. Установите на экране анализатора кадр МНЕМОНИКА. На экране анализатора должно появиться изображение в соответствии с табл.3.

Таблица 3

Сбор данных в мнемоническом коде

БАЗА	16	МНЕМОНИКА	
	АДРЕС	КОП	ДАННЫЕ
01	0000	MVI	A,92
03	0002	OUT	FB
05	FBFB	92	ВЫВОД
06	0004	JMP	003B
09	003B	MVI	A,FF
11	003D	OUT	FA
13	FAFA	FF	ВЫВОД
14	003F	MVI	A,8F
16	0041	OUT	FA
18	FAFA	8F	ВЫВОД
19	0043	LXI	H,83EC
22	0046	MVI	B,OC
24	0048	XRA	A
25	0049	MOV	M,A
26	83EC	00	ЗАПИСЬ
27	004A	INX	M
28	004B	DCX	B
29	004C	JNZ	0049
30	0049	MOV	M,A
33	83ED	00	ЗАПИСЬ

Это начало программы-монитора "Микролаб КР580ИК80".

Урок 19

ДИАГНОСТИКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Принципы диагностики микропроцессорных систем в основном не отличаются от принципов диагностики обычных цифровых устройств. Вначале необходимо изучить принципиальную схему и правила работы по структурной схеме. В этом уроке описываются проблемы, относящиеся к микропроцессорным системам и методам их диагностики.

Существуют некоторые проблемы, характерные только для микропроцессорных систем. Большая часть управления реализована программным способом, поэтому трудно следить за распространением сигнала. Микропроцессорные системы в отличие от многих логических схем не могут исследоваться в статическом режиме. Измерения должны быть выполнены в процессе работы МП, что ограничивает

эффективность использования логического пробника и стимулятора.

Дополнительные трудности создают шинные структуры микропроцессорных систем. Данные на этих шинах зачастую не стабильны или не имеют смысла вследствие наличия выходов, имеющих три состояния, мультиплексирования и переходных процессов, возникающих при переключении. Это не вызывает затруднений в работе системы, так как она синхронна и ей известно, когда на шине данные стабильны. При использовании сигнатурного анализатора и анализатора логических состояний при соответствующем выборе сигналов синхронизации, нетрудно контролировать сигналы на линиях шин в моменты, соответствующие их стабильному состоянию. Осциллограф может быть использован для проверки логических уровней, временных соотношений импульсов состязания в шинах и определения общей активности схемы.

Вследствие того, что шинная структура позволяет к одной точке подсоединять множество устройств, обнаружение одного дефектного устройства в таком узле затруднительно. Шина данных,кроме того,действуя как цепь обратной связи цифрового сигнала, способствует распространению ошибки через правильно работающую схему и возвращению ее снова к источнику ошибки. Лучшим способом решения этой проблемы является разрыв цепи обратной связи.

К шинам МП часто подсоединяются сложные устройства. Эти устройства трудно проверять с использованием метода стимулов/откликов. Правильность работы этих устройств может проверяться путем замены их заведомо исправным устройством или контроля правильности их функционирования в системе.

МП являются последовательностными устройствами. Выполнение программы МП зависит от всей последовательности команд и событий. Если неправильным является хотя бы один бит информации, то может произойти сбой работы всей системы. Шумовые "всплески" и дефектные разряды ЗУ являются наиболее распространенными источниками однобитовых ошибок. Подобные неисправности очень трудно локализовать, так как происходит сбой системы в целом.

Контролепригодная конструкция и хорошая документация, использование сигнатурного анализатора и других устройств значительно облегчают диагностирование.

"Микролаб" является типичной микропроцессорной системой.Благодаря тому, что "Микролаб" специально разработана с учетом возможности обучения диагностированию, ее принципы могут применяться к различным классам микропроцессорных систем.

ПРОБЛЕМЫ ДИАГНОСТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Импульсы такта

Неправильные импульсы такта могут вызвать сбой в работающей системе. Некоторые МП чувствительны к величине частоты тактовых импульсов. Так как многие системы должны работать в соответствии с техническими требованиями, даже небольшое отклонение от заданного значения частоты тактовых импульсов /слишком большая частота/ может вызвать сбой системы. При медленной работе системы сбой происходит вследствие неправильной работы ячеек динамической памяти. Возникновение подобных ситуаций более вероятно при использовании схем синхронизации на резисторах и конденсаторах /RC/ вместо более точных и стабильных схем на основе кварцевых резонаторов. Вместе с тем, кварцевые резонаторы иногда могут переходить в генерацию на третьей субгармонике, вызывающей значительно большую, чем ожидалось, частоту тактовых импульсов. Кроме того, для некоторых МП требуются неперекрывающиеся тактовые импульсы, имеющие очень строгие ограничения временных соотношений. Уровни напряжения тактовых импульсов могут быть не всегда совместимы с ТТЛ-уровнями, должны иметь значительно большие уровни напряжения. Характеристики тактовых импульсов приводятся в технических требованиях на МП и могут проверяться с использованием обычных частотомеров и осциллографов.

Сброс при включении питания

Схема сброса при включении питания может вызвать сбой работающей системы. Импульсы сброса, не соответствующие техническим требованиям /слишком короткие или с большими помехами, или имеющие большие длительности фронта и спада/, могут дать неправильную начальную установку прибора, что приводит к ошибочной работе, частичной или полной /отсутствие функционирования/. Сбои могут возникать вследствие срабатывания схем сброса от паразитных выбросов источников питания при слишком большой чувствительности этих схем. Длительные фронты могут вызвать разброс синхронизации сброса у разных блоков системы даже при использовании схем Шмидта. Это приведет к несинхронизированному подключению питания различных блоков, что приводит к сбою работы всей системы. Повторное включение питания прибора при слишком короткой продолжительности выключенного состояния может привести большинство систем /в том числе и "Микролаб"/ к неправильному запуску. Может возникнуть необходимость увеличить время состояния ВЫКЛЮЧЕНО, чтобы позволить восстановиться источникам питания и схемам сброса.

При этом необязательно, что подобные дефекты приводят к останову системы. Она может некоторое время поработать, а затем остановиться или зациклиться, или даже выполнить большую часть нормальных операций. Необходимо помнить, что система должна полностью завершить последовательность сброса при включении питания, что гарантирует выполнение всех операций проверки, управления и начальной установки, необходимых для включения системы.

Схемы сброса при включении питания работают при включении системы. В это время их можно наблюдать с помощью осциллографов, имеющих ЗУ, анализаторов логических состояний и в отдельных случаях сигнатурных анализаторов. Для испытательных целей они могут вручную повторно запускаться и управляться извне.

ПРЕРЫВАНИЯ

Линия прерывания при помехах или "залипании" может вызвать сбой работы системы. Система может работать и с "залипшей" линией, но очень медленно /большая часть времени уходит на обработку мнимых прерываний/. Помехи в линии прерывания могут вызвать непредусмотренные изменения в системе, например, ввод/вывод периферийных устройств выполняется в неуказанное время или система перестает реагировать на отдельные устройства ввода/вывода.

Активность линии прерывания может проверяться логическим пробником, анализатором логических состояний или осциллографом. Прерывания могут быть асинхронными,а для испытательных целей они могут управляться вручную /разрешаться либо запрещаться/.

Взаимные искажения сигнала

В длинных параллельных шинах и линиях управления в микропроцессорных системах достаточно больших размеров возможны взаимные помехи особенно в той части, где проходят синхроимпульсы и сигналы разрешения. Результатами подобных помех являются большие выбросы на соседних сигнальных линиях или "звон" на запускающей линии. При этом могут вводиться ошибочные данные или управляющие сигналы, которые очень трудно обнаружить. Эта проблема чаще всего возникает, когда сигнальные линии длинны и нарушены допустимые нормы временных характеристик сигналов и шума в системе.

Дефекты могут возникать, когда эти системы дополняются платами-удлинителями или находятся в условиях повышенной влажности. Взаимное расположение проводников может вызвать появление дефекта, когда быстродействующие линии передачи сигналов /такие как выходные сигналы вентилей с диодами Шоттки/ идут вдоль других сигнальных линий, даже если они расположены на противоположной стороне печатной платы.

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Дефекты памяти в микропроцессорной системе могут вызвать сбой в работе системы различного характера - от ошибки в одном бите запомненных данных до сбоя работы всей системы. Большинство дефектов ЗУ может быть обнаружено во время программы самопроверки при включении питания, если дефекты памяти не разрушили этой программы. Если система не обеспечивает проверки ОЗУ и нет сервисного разъема, то практически его невозможно проверить. В этом случае необходимо заменить подозреваемое дефектное ОЗУ.

Дефекты ОЗУ, возникающие в области памяти, используемой в качестве стековой памяти, обычно вызывают сбой системы, даже если дефектным оказался 1 бит. Схема регенерации динамического ОЗУ является другим фактором, который надо учитывать при диагностике явных дефектов ОЗУ.

ПЗУ также могут оказаться дефектными. Такие дефекты возникают чаще всего при использовании устройств, программируемых без маски. Один неправильный бит может вызвать сбой системы. ПЗУ может эффективно проверяться во время самопроверки при включении питания, если такой режим предусмотрен. Но в отличие от ОЗУ, ПЗУ может также проверяться с помощью и других методов /если отсутствует самопроверка/. При одном из таких методов используется режим свободной работы системы и сигнатурный анализатор для проверки записанных в документации сигнатур и для сравнения выходов подозреваемого и эталонного ПЗУ.

ПРОГРАММНЫЕ САМОПРОВЕРКИ

Возможность программирования микропроцессорных систем облегчает проверку. Программы, хранящиеся в ПЗУ системы, могут проверять ПЗУ, ОЗУ и сам процессор. Часто можно проверить и устройство ввода/вывода. Программное обеспечение может быть использовано для выработки стимулов при использовании диагностического устройства, например, сигнатурного анализатора.

ПРОВЕРКА ПЗУ

Наиболее распространенным методом проверки ПЗУ является метод контрольных сумм. Когда ПЗУ запрограммировано, содержимое всех ячеек складывается, переносы игнорируются. Полученный результат в дополнительном коде заносится в последний /иногда в первый/ адрес ПЗУ. Поэтому при сложении всех слов, /включая контрольную сумму, запомненную в последнем байте/, получаем нуль. Если в конце испытательной последовательности сумма не равна нулю, то ПЗУ имеет дефект /на практике контрольная сумма обычно вычисляется с целью получения в итоге определенного числа, отличного от нуля/.

Метод контрольных сумм позволяет обнаруживать одиночные однобитовые и большинство многобитовых ошибок. Но существует много различных комбинаций с двумя и более ошибками, дающих правильную контрольную сумму. Таким образом, ПЗУ, прошедшее тест методом контрольной суммы, возможно является исправным с какой-то вероятностью. Если же тест не проходит, значит существует дефект /хотя он может быть не в самом ПЗУ/.

Проверка ОЗУ

Для проверки ОЗУ осуществляется запись последовательности в память, считывание ее и выявление изменений. Наиболее распространенной является шахматная последовательность, состоящая из чередующихся 1 и 0. После проверки всех ячеек памяти последовательность повторяется в обратном порядке. Проверка гарантирует, что каждый бит ОЗУ может запоминать 1 и 0. Но не все испытательные тесты для ЗУ могут гарантировать 100%-ную вероятность обнаружения дефекта. Даже в том случае, если каждая ячейка памяти может запомнить 1 или 0, ОЗУ могут быть чувствительны к самой последовательности. Например, какая-то одна ячейка может запомнить 01010101 и 10101010, но дефект может проявиться, если запоминается 01111000. Даже в небольших ОЗУ для проверки каждой возможной комбинации потребуется много времени. По этой причине надежность проверки ОЗУ значительно ниже надеж-

ности проверки ПЗУ. Как и при тесте методом контрольной суммы, если ОЗУ проходит программу самопроверки, то оно считается годным с какой-то вероятностью. Если не выполняется тест, это говорит о наличии дефекта.

Мультиплексированный ввод/вывод

Мультиплексированные клавиатуры и дисплеи часто имеют общую схему сканирования. В этом случае "залипание" клавиши вызывает сбой дисплея. Подобным образом неисправность индикатора вызовет сбой клавиатуры. Для выполнения диагностики необходимо учитывать взаимодействие между схемами общего сканирования.

Интерфейсы

Многие микропроцессорные системы сопрягаются с другими системами через внешние линии связи. Эти линии часто бывают длинными, на них оказывают влияние источники электрических помех: реле, трансформаторы, моторы, соленоиды и даже молнии. Эти электромагнитные помехи могут вызвать передачу неправильных данных, перегрузку схемы интерфейса и значительные повреждения компонентов /в случае молнии/. В подобных случаях, а также при наличии больших переходных токов, являющихся следствием емкости длинных кабелей интерфейса, схемы драйверов выходных линий имеют тенденцию к увеличению частоты появления дефектов.

Схемы алгоритмов диагностирования

Схемы алгоритмов диагностирования - это графическое средство показа последовательности проверок, выполняемых в проверяемом приборе. Результаты каждой проверки определяют последующий шаг. Использование схем алгоритмов диагностирования при ремонте микропроцессорных приборов экономит время и труд.

На рис.1. показана часть схемы алгоритма диагностирования цифрового вольтметра. Теоретически эта схема должна определить дефект прибора, но на практике это не всегда так. Совершенная схема алгоритма диагностирования должна учитывать

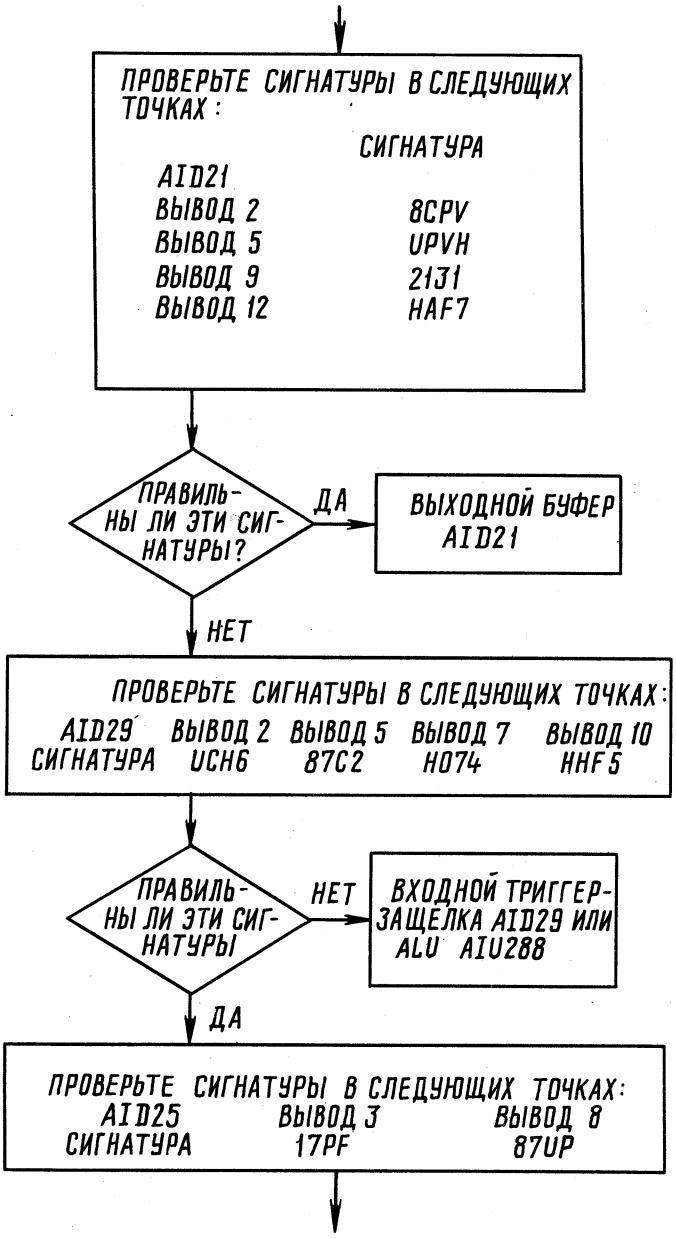


Рис.1. Типичная схема диагностирования с помощью анализатора сигнатурного

все возможные дефекты, что значительно усложняет разработку этой схемы. Кроме того, диагностические схемы имеют общий характер, в них недостаточно определенности, необходимой для выполнения проверки и принятия решения. Некоторые схемы диагностики дают практическую информацию о том, как определенный тест или измерение соотносятся с тем, что выполняет или должна выполнять схема. Если схема диагностики не приводит Вас к фактическому дефекту, то Вы не будете знать, что дальше делать. Однако гораздо чаще схемы диагностики будут служить Вам руководящим материалом.

Существуют хорошие и плохие схемы диагностирования. Первые обеспечивают логичную, хорошо ориентированную последовательность проверок и измерений, требующих минимального понимания проверяемой схемы. Часто они включают такие современные методы, упрощающие процедуру, как сигнатурный анализ. При диагностике прибора даже не очень хорошие схемы диагностирования могут быть полезны для локализации неисправной области в системе.

Другие документы

Опытный инженер-разработчик из структурной схемы прибора может получить достаточную информацию о том, как вместе работают различные части схемы. Но теория работы прибора и схемы его диагностирования не дают детального представления об аппаратной части. Принципиальная схема включает в себя слишком много детальной информации, что затрудняет наблюдение работы в целом.

Существует ли неисправность на самом деле

Важно иметь общее представление о приборе, чтобы быть уверенным в реальности существования неисправности. В определенной степени необходимо знать, что и как делает прибор. МП позволяют разрабатывать приборы, способные выполнять определенные функции,но в то же время и являются причиной трудностей при их обслуживании. Необходима уверенность в том, что неисправность реально существует в приборе. Нет ничего бесполезнее попытки исправить то, чего на самом деле не существует. В некоторых ситуациях оказывается, что от прибора хотят выполнения того, для чего он не был предназначен. Например, переключатель выбора уровня сигнала цифрового вольтметра может работать, показывая вольты, но не амперы. Такое "ограничение конструкции" обычно может быть обнаружено при чтении инструкции по эксплуатации и является не дефектом прибора, а лишь его недостатком.

Ошибки при разработке программной части /ПЗУ/ могут иногда вызвать сбой при работе в условиях, не учтенных во время проектирования. Их появление наиболее вероятно на ранних стадиях изготовления. С другой стороны, проблема неисправности действительно может существовать, но не проявляться, так как прибор проверяется неадекватно. Проблему такого типа часто определить довольно просто /например, индикатор Омов гаснет, когда на цифровом вольтметре нажимается кнопка "Ом"/.

Проблема наличия неисправности может быть довольно сложной. Например, ошибки могут возникать при выполнении необычной последовательности операций. Из-за того, что весь комплекс проблем довольно сложно выявить, для проверки приборов в производственных условиях используется множество испытательных процедур.

Что можно увидеть на передней панели

Даже не снимая крышку прибора, можно получить достаточно много диагностической информации. Большинство приборов, построенных на основе МП, имеют переднюю панель, на которой могут находиться переключатели, индикаторы, устройства ввода/вывода. Работа с передней панелью - это процесс, при котором переключатели, кнопки и другие устройства ввода используются для получения "откликов" от прибора, наблюдаемых на индикаторах или других выходных устройствах.Например, если индикаторы не зажглись при включении питания, то можно предположить наличие неисправного переключателя, плавкого предохранителя, шнура питания, подсоединения батарей или источника питания. Если не светится один сегмент дисплея, то вероятно неисправен сам индикатор или схема, запускающая его. Если единственный дефект цифрового вольтметра проявляется только в диапазоне входного сигнала 1-10 В, то область дефекта может сузиться до относительно небольшой части схемы /аттенюатор/.

Необходимо всегда использовать преимущества встроенных режимов проверки характеристик и проверки при включении питания, а также

имеющихся указаний по диагностике в рабочей инструкции.

Что можно узнать из инструкции

"Если появился дефект - смотрите в инструкцию".

В инструкции находится огромное количество советов. Работу по диагностике могут значительно облегчить специальные сервисные переключатели, переходники, подключающие устройства, индикаторы и специальные испытательные методы.

Необходимо изучить: принцип работы прибора, структурную и принципиальную схемы. Не нужно это делать слишком подробно, достаточно понять общие идеи. Определите местоположение МП, ПЗУ, ОЗУ, устройств ввода/вывода, адресного дешифратора, синхрогенератора, шины управляющей части и части прерывания в системе.

Сравнение дефектов, появляющихся в процессе производства, с дефектами, возникшими в процессе эксплуатации

Работа интегральных схем /ИС/ обычно проявляется как последовательность определенных событий. ИС создается на заводе-изготовителе интегральных схем и посылается изготовителю приборов. Там микросхема устанавливается на печатную плату, которая вставляется в прибор. Затем прибор поступает к потребителю. Изготовители приборов считают, что $\sim 2\%$ поступающих интегральных схем дефектны. Стоимость обнаружения дефектной микросхемы на этапе входного контроля значительно ниже, чем после того, как микросхемы установлены на печатные платы.Если неисправные микросхемы не обнаружены до полной сборки прибора, то диагностика неисправности и ремонт на заводе возрастают примерно в 10 раз. Замена дефектной микросхемы на месте эксплуатации возрастает еще в 10 раз. Таким образом, экономически выгодно обнаружить дефектные схемы как можно раньше.

Общие источники дефектов и методы наилучшего их обнаружения зависят от предыстории изготовления прибора и условий, в которых он проверяется. Когда новый прибор на заводе включается впервые, у него почти всегда что-нибудь оказывается неисправным. Те приборы, которые оказываются дефектными на месте эксплуатации, когда-то были неисправными. Дефекты сборки, такие как компоненты, неправильно нагруженные и запаянные, в общем случае не относятся к дефектам, появляющимся при эксплуатации.Вероятность появления коротких замыканий при пайке и многократных дефектов гораздо больше на производстве, чем на месте эксплуатации. Непроизводственные дефекты обычно вызваны дефектными компонентами или соединениями.

АВТОМАТИЧЕСКИЕ ТЕСТЕРЫ

Специализированное испытательное и диагностическое оборудование применяется для проверки большого количества одинаковых приборов.

Для минимизации времени обнаружения дефектов часто используются автоматические тестеры печатных плат и подключающие устройства. Они обеспечивают быструю, экономичную и точную проверку и диагностику дефектов. Но нельзя слишком доверять автоматическим тестерам плат, управляемым ЭВМ. Иногда фактически неисправные платы, выдерживают проверку тестером, что является следствием недостатка самого тестера /синхронизация, нагрузка, проверка компонентов/ или выполняемой испытательной программы. Тем не менее, современные тестеры плат, выполняющие более сложные динамические, функциональные, параметрические тесты, в общем заслуживают доверия.

Что легче проверять

Сначала необходимо проверить те части схем, которые легко проверяются и восстанавливаются. Вероятность сбоя простых и сложных схем одинакова. Обычно в большинстве приборов сбой чаще всего появляется в источнике питания. Его легче всего проверять и диагностировать. Напряжение, не соответствующее требова-

ниям технических условий, может привести к неверной работе схем. Если, в первую очередь, не проверить напряжение, то для обнаружения неисправности может потребоваться значительное время.

Полезной может оказаться визуальная проверка конструкции. Визуально или на ощупь можно определить плохое подсоединение печатных плат и кабелей, разорванные провода, отсутствующие компоненты.

Общие проблемы диагностики, возникающие в процессе производства

При тщательной визуальной проверке схем прибора можно обнаружить немало общих источников дефектов. Легко обнаружить неправильно установленные переключатели и перемычки, неправильно нагруженные компоненты /неправильно поставленные/, мостики холодной пайки. Неправильно установленные резисторы очень трудно обнаружить электрически, так как они могут вызвать взаимодействие между не связанными по схеме логическими узлами, но визуально эту ошибку определить легко.

Наиболее общим производственным дефектом являются короткие замыкания между проводниками на печатных платах. Обычно их можно легко устранить острым ножом. Когда точное место короткого замыкания неизвестно, можно использовать достаточно новый метод, применяемый также в случаях, когда место короткого замыкания недоступно /таких как короткие замыкания внутренних слоев многослойных плат/. Процедура заключается в следующем: зарядите конденсатор 100 мкФ /или больше/ до напряжения 5 В /безопасное напряжение для логических схем/. Затем с соблюдением правильной полярности подсоедините его с помощью кабеля /проводов/ к плате. Конденсатор разряжается. На плате слышен характерный звук. Проверьте наличие короткого замыкания. Если оно не разрушилось, необходимо повторить процедуру. Этот метод надо использовать осторожно, так как обнаруживается самое слабое звено в ветви тока, которое не всегда может оказаться источником дефекта, а лишь только тонким местом в проводнике или на плате.

Относительно новой проблемой являются микросхемы с подогнутыми выводами, появившиеся в результате применения автоматического оборудования для установки компонентов. Это может привести к отсутствию электрического соединения между микросхемами и печатной платой, неправильному соединению или короткому замыканию с проводниками, находящимися около или под микросхемой. Подогнутый вывод микросхемы часто трудно обнаружить визуально, поскольку он выглядит так, как будто припаян правильно. Лучше всего проверить на нижней стороне платы наличие концов выводов микросхемы или посмотреть под самой микросхемой.

В приборе обычно используются краевые разъемы для печатных плат. Они могут вызвать появление дефектов, когда их контакты отходят от печатной платы, изолируя тем самым плату от разъема, или когда в контакты разъема случайно попадает проводник. Эти дефекты также можно обнаружить при визуальной проверке.

Многослойные платы имеют и некоторые специфические особенности. Например, если плату посмотреть на свет, то можно обнаружить разрушение внутренних слоев /которое может быть вызвано высокой частотой сигнала или током утечки/. Из-за того, что ремонт внутренних слоев невозможен, приходится заменять всю плату.

На платах с использованием монтажа накруткой могут сгибаться штырьки, что может вызвать короткое замыкание. Другими распространенными дефектами являются: неправильная установка 14-выходной микросхемы в 16-выводное гнездо, неправильная пайка, короткое замыкание между выводами; объединение сигналов /взаимные помехи/ вследствие близкого расположения проводников.

МЕХАНИЧЕСКИЕ ДЕФЕКТЫ, ВОЗНИКАЮЩИЕ ПРИ ЭКСПЛУАТАЦИИ

При визуальной проверке изделий, оказавшихся дефектными при эксплуатации, можно обнаружить: разорванные проводники, сломанные керамические корпуса микросхем и

корпуса резисторов, согнутые штырьки-накрутки и загрязненные пластины контактов. Часто для исправления "залипшей" контактной пластины разъемов и реле достаточно постучать по стенке разъема или реле. Подобные неисправности возникают при механическом сжатии плат и разъемов. Если прибор не работает сразу после поступления или дает сбой в экстремальных условиях, то можно предположить, что краевой разъем печатной платы дефектен. Если есть подозрение, что причиной дефекта является плохой контакт разъема, то необходимо просмотреть все соединения блоков и плат. Для очистки загрязненных краевых разъемов можно использовать специальный стирающий карандаш или смесь для промывки печатных плат.

ОБЩИЕ МЕТОДЫ ДИАГНОСТИКИ

Замена плат

Если печатная плата легко вынимается и имеется запас исправных плат, то можно попытаться заменить ее. Когда в одном приборе используется несколько одинаковых плат, то их можно взаимно заменить. Недостатком этого метода является опасность повредить годную плату вследствие электрических перегрузок, которые повредили предыдущую плату. В любом случае при выполнении смены платы питание от прибора должно отключаться. Если имеются аналогичные изделия, то, сравнивая их функционирование, можно получить достаточно много информации. Это сравнение может быть полезным в ситуациях, когда неясно, существует ли действительно дефект аппаратной части /это может быть общим недостатком самого прибора или его конструкции/.

Если подозреваемое устройство находится в гнезде разъема, то нужно его сначала покачать,чтобы удостовериться в отсутствии оторванных соединений, а затем его можно заменить. Фактическая частота отказов МП очень невелика. Тем не менее, из-за сложности проверки правильности их работы чаще всего их первыми вынимают из платы. Это характерно также и для кристаллов БИС, используемых вместе с ними.

Проверка воздействием /СТРЕСС-ПРОВЕРКА/

СТРЕСС-ПРОВЕРКА может быть очень эффективной для обнаружения промежуточных или перемежающихся отказов. СТРЕСС-ПРОВЕРКА часто может вызвать временное исправление или ухудшение дефектов /любой из случаев полезен для обнаружения места дефекта/. На платы оказывают физическое воздействие, нагревают или охлаждают, изменяют напряжение питания. Термические воздействия могут использоваться для локализации дефекта в определенном устройстве на плате с гораздо большей точностью, чем при других методах, так как тепло или холод может прикладываться непосредственно к одиночному компоненту. Перемежающиеся дефекты могут возникать вследствие критичных кристаллов, спутанных проводников, мостиков паек, соединений, схем запуска и синхронизации.

Легкое прикосновение к каждому устройству на плате помогает выявить нагревающийся компонент /более горячий, чем соседние компоненты/. Когда какой-то компонент нагревается больше других компонентов того же типа, то, вполне вероятно, что он неисправен. Дефектное устройство может быть настолько горячим, что можно обжечься, поэтому необходимо соблюдать осторожность. Надо также иметь в виду, что некоторые вполне годные устройства могут нагреваться больше нормы.

Короткие замыкания источников питания

Существует несколько эффективных способов обнаружения причины короткого замыкания источника питания. В системе, имеющей много плат, первое, что необходимо сделать, - это локализовать короткое замыкание с точностью до одной платы. Для этого платы вынимаются по очереди до тех пор, пока в источнике не исчезает короткое замыкание. Последняя вынутая плата является причиной короткого замыкания источника питания.

Одним из методов обнаружения короткого замыкания в шинах питания является обеспечение сравни-

тельно высокого тока /3-5 А/ в этих шинах. Необходимо соблюдать полярность подключаемых источников и не превышать номиналов обычного напряжения питания. Ответвление тока, ведущее к короткому замыканию, может быть определено с помощью цифрового вольтметра, имеющего высокое разрешение /0,01 мВ/, который позволяет определить падение напряжения на отдельных участках шины питания. Напряжение будет присутствовать лишь в частях шины питания, ведущих к короткому замыканию /рис.2/.

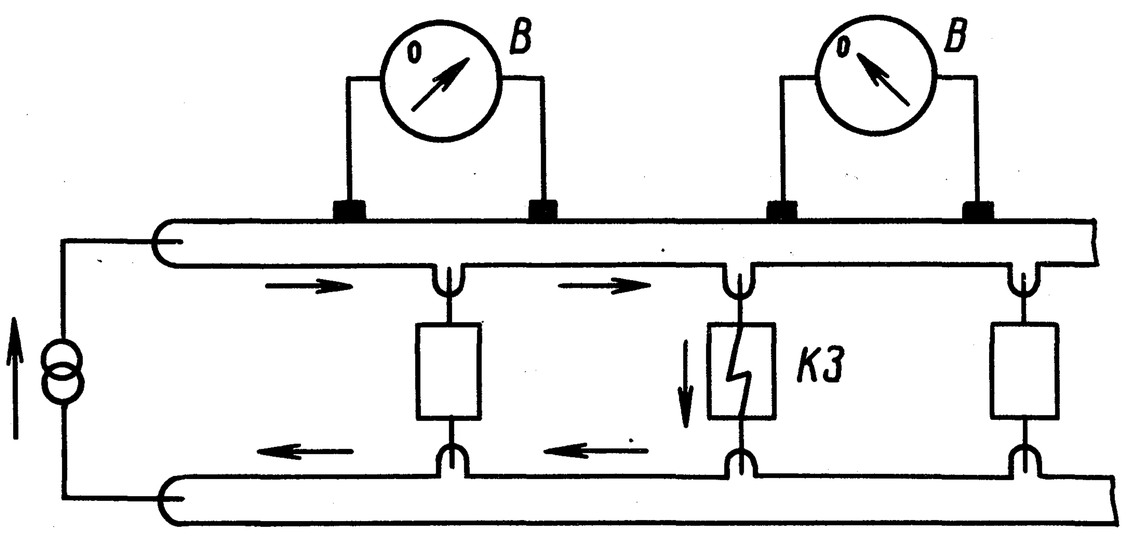


Рис.2. Использование чувствительного вольтметра для обнаружения короткого замыкания шины питания

Более наглядным способом обнаружения короткого замыкания шин питания /хотя и менее приемлемым в реальной жизни/, является замораживание всей платы /примерно до -10°С/, при этом жидкость конденсируется на плате, затем плата подключается к источнику тока 3-5 А. По мере нагревания и размозки платы ветвь тока становится видимой. После этого легко обнаружить короткое замыкание.

КАК ЛОКАЛИЗОВАТЬ ДЕФЕКТ

После выполнения наиболее легко осуществляемых проверок нужно переходить к более сложным. Теперь имеет значение индивидуальный опыт по отысканию неисправностей, интуиция и знание прибора. Во-первых, необходимо использовать все преимущества встроенных и описанных в документации возможностей изоляции неисправностей в схеме, таких как выборочное изъятие отдельных плат, сервисные перемычки, специальные тестовые режимы и процедуры. Также очень полезно отсоединять микропроцессорную систему от периферийных схем, что позволит диагностировать каждую часть независимо.

Важным способом диагностирования является метод деления на половины, который заключается в выборе какой-то точки приблизительно в середине схемы. Имеется вероятность того, что дефект находится до или после этой точки. Если характеристики схемы правильны до этой точки, то дефект находится после нее. Если нет, то находится перед этой точкой. Этот метод лучше использовать на схемах с четкими, однонаправленными сигнальными ветвями без больших ветвей с обратными связями. Этот метод может быть эффективен даже в микропроцессорных системах, так как схемы, не входящие в микропроцессорную часть, могут быть проверены таким способом.

В типичном изделии первое разделение обычно выполняется на цифровую и аналоговую части, если таковые есть. Аналоговые схемы имеют больше дефектов /из-за высоких требований, предъявляемых к быстродействию, питанию, температуре, чувствительности, точности, регулировкам, внешним избыточным нагрузкам и допустимому разбросу характеристик компонентов/. Сбой прибора часто возникает из-за аналоговых схем, работающих на пределе. Они могут вывести из строя и цифровые схемы. Необходимо учитывать электрические связи цепей синхроимпульса и шин питания ТТЛ-схем с аналоговыми цепями, которые могут вызвать серьезные помехи в системе.

ВИДЫ ДЕФЕКТОВ ЦИФРОВЫХ СХЕМ

Если есть подозрение, что цифровая часть неисправна, необходимо сначала убедиться в наличии сигналов. Логическим пробником можно проверить наличие сигнала в цепи синхроимпульса, на линиях шин, в цепях разрешения кристалла и управляющих линиях. Отсутствие активности в любом из этих узлов указывает на наличие неисправности.

Наиболее распространенным видом отказа цифровых микросхем является разрыв проводников внутри кор-

пуса. Имеются тонкие проводники, соединяющие выводы корпуса с кристаллом микросхемы. Если выходной проводник разрывается, то выходной вывод оказывается изолированным, и логический пробник, возможно, укажет на постоянный средний логический уровень, поскольку входы других устройств подсоединены к этому узлу. При разрыве входного проводиника один или более выходов микросхемы окажутся неисправными /"залипание" на 1, 0 или неправильное выполнение логической функции/. Если любой из этих выходов переходит на общую шину, имеющую три состояния, это может вызвать появление неисправности /более одного выхсда в данный момент/. Неисправности шин можно наблюдать на осциллографе, они индицируются стабильными неправильными логическими уровнями шин, но осциллограф не выдает информации об источнике дефекта /рис.3/, так как когда все устройства на шине выключены, то и исправные линии шин могут иметь низкие логические уровни.

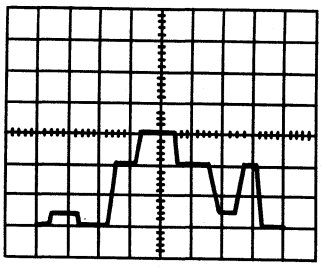


Рис.3. Пример неправильных /но четких/ логических уровней, вызванных неисправностями в шинах

Другим распространенным дефектом в цифровых микросхемах является короткое замыкание между входным выводом и землей. Этот дефект часто вызывается неисправностью защитного диода на входе кристалла. Этот дефект обычно проявляется как "залипший" низкий уровень, который можно наблюдать логическим пробником. Осциллограф, подсоединенный к такому узлу, покажет уровень напряжения немного выше уровня корпуса, порядка нескольких сотен мВ, во время подачи на этот вход уровня лог. 1. /рис.4./.

Для обнаружения "залипших" входов и выходов может использовать-

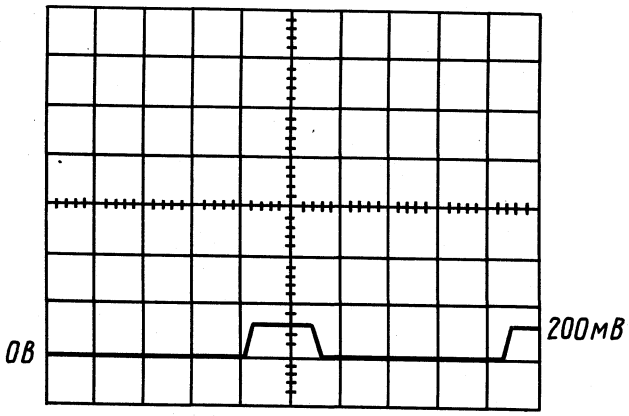


Рис.4. Логический уровень, вызванный коротким замыканием входного контакта микросхемы на корпус /короткое замыкание подложки диода/

ся чувствительный цифровой вольтметр /с высоким разрешением/ и охлаждающая аэрозоль. Подсоедините цифровой вольтметр к "залипшему" узлу и выберите наибольшую чувствительность вольтметра в режиме постоянного тока. Затем, во время наблюдения напряжения по очереди обрызгивайте каждую микросхему,подсоединенную к "залипшему" узлу, тем самым изменяя их температуру. Любое обнаруженное изменение величины напряжения /более 10 мВ/ в узле показывает, что "опыленная" микросхема является потребителем тока.

Если охлаждение невозможно, то можно использовать нагревание. Этот метод основан на свойствах полупроводников, используемых в ИС /связь между напряжением и температурой/.

МЕТОДЫ ЛОКАЛИЗАЦИИ

Подозреваемый входной или выходной контакт микросхемы необходимо изолировать от остальной части схемы. Быстро и надежно это можно выполнить путем снятия припоя, находящегося между выводом и печатной платой с помощью паяльника с воздушным отсосом с последующим обметанием щеточкой. Затем нужно подогнуть вывод так, чтобы он не касался краев отверстия платы. С помощью тестера удостоверьтесь, что вывод не

имеет электрического контакта с платой.

Выбор метода, который можно использовать для изоляции цифровых блоков МП, полностью зависит от их электрической и механической архитектуры. Если после отклонения некоторых цифровых плат ядро продолжает работать, то такая процедура будет полезна при отыскании неисправностей. Если ядро может работать с разомкнутой петлей /при отсутствии обратной связи по шине данных/, то для проверки активности ядра и адресных шин может использоваться свободный режим.

Для разрыва цепей отдельных сигналов между печатной платой и остальной частью системы может использоваться плата удлинителя с переключателями на шине и сигнальных линиях. Таким же образом в цепи обратной связи "залипшие" шины могут отключаться от основной системы.

Более простым способом разрыва отдельных цепей, идущих через краевой разъем, является изоляция контактов разъема от платы с помощью изоляционной ленты или жесткой бумаги. Необходимо помнить платы, на которых это было проделано, чтобы потом не забыть убрать изоляцию. Менее удобным, но более эффективным средством определения неисправностей в шине является измерение сопротивления /при отключенном питании/ между каждой линией шины и землей в отдельности, например, в шине данных и в адресной шине.

Сопротивление каждой из этих линий обычно одинаково. Если в одной из линий сопротивление значительно отличается, то она, возможно, является дефектной. Если в двух линиях сопротивления одинаковы и ниже, то они могут быть закороченными. В любом случае сначала проверьте схему, чтобы понять, обусловлена ли эта разница сопротивлений организацией схем, подсоединенных к этим линиям.

При проверке правильности работы подозреваемой микросхемы могут использоваться линии прерываний и выводы выбора кристалла. Это осуществляется моментальным закорачиванием соответствующего вывода на высокое или низкое состояние или с помощью логического стимулятора.

Петли обратной связи

Цифровые схемы, охваченные обратной связью, трудно поддаются диагностике, так как дефекты распространяются по кругу. В петле обратной связи неправильный выходной сигнал поступает снова на вход, вследствие чего везде получаются неправильные результаты. Размыкание этой цепи обратной связи исключает поступление на вход неправильных выходных сигналов. Затем /если возможна генерация управляющих входов сигналов для этой петли/ можно наблюдать распространение сигналов с входа на выход. Тем не менее, часто трудно обеспечить необходимые входные воздействия, так как нужно управлять множеством линий. Если имеется другой аналогичный и действующий прибор /или плата с такой же схемой/, то можно управлять входами обеих схем с выхода годной схемы. В этом же случае есть уверенность, что проверяемая схема получает правильный входной сигнал. Затем нужно только сравнивать соответствующие узлы двух схем и определять различие между ними. Для выполнения этого сравнения можно использовать сигнатурный анализатор.

Другой метод локализации дефектной микросхемы заключается в следующем: выходы подозреваемой микросхемы контролируются осциллографом или сигнатурным анализатором, сверху запаивают идентичную микросхему /за исключением выходных контактов/.

Выводы верхней микросхемы следует слегка подогнуть /если необходимо, они все приводятся в контакт/. Изменение сигнала может указать на наличие дефекта этого устройства. Если не наблюдаются изменения и выход не является "залипшим", можно считать, что микросхема не дефектна. С последовательностными схемами, такими как счетчики и регистры сдвига, необходимо соблюдать осторожность, в них отклонение в выходном сигнале может возникнуть из-за начальных условий. Лучшим способом выполнения этого теста является использование компаратора интегральных схем.

ЗАКЛЮЧЕНИЕ

Никакие знания и опыт не могут полностью компенсировать несовершенную документацию по обслуживанию. Разрабатываемые приборы должны быть ориентированы на современные методы обслуживания, такие как сигнатурный анализ, так как старые методы и устройства, предназначенные для диагностики обычной логики, недостаточно эффективны при работе с МП.

Микропроцессорные системы являются следствием развития традиционной цифровой логики. Многие компоненты, принципы построения схем, средства и методы их диагностики одинаковы. Тем не менее имеются и некоторые различия. Микропроцессорные системы имеют шинную структуру, а многие устройства на шине являются сложными БИС. В шинах между устройствами постоянно присутствуют сложные сигналы. Чтобы изолировать дефект, вызвавший сбой всей системы, часто полезно прервать шину данных,являющуюся основной ветвью обратной связи системы.

Хотя схемы алгоритма диагностирования и обеспечивают правильный порядок локализации дефектов изделий, этого оказывается недостаточным для полной локализации неисправного компонента. Существует множество методов и процессов, эффективных при выполнении диагностики, изоляции и локализации дефектов изделий на основе МП.

Приложение

НАБОР КОМАНД МИКРОПРОЦЕССОРА КР580ИК80

Все команды МП КР580ИК80 делятся на пять групп.

1. Команды пересылки данных. Осуществляют пересылку данных между регистрами или между памятью и регистрами.

2. Арифметические команды. Предназначены для выполнения сложения, вычитания, увеличения или уменьшения содержимого регистров или ячеек памяти.

3. Логические команды. Выполняют логические операции "И", "ИЛИ", "исключающее ИЛИ", сравнение, сдвиг и дополнение данных в регистрах и ячейках памяти.

4. Команды переходов /условных и безусловных/, вызова подпрограмм и возвращения из подпрограмм.

5. Команды ввода/вывода, управления и работы со СТЕКОМ. Предназначены для выполнения операций ввода/вывода, работы со СТЕКОМ, управления флагами, разрешения и запрещения прерываний.

КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

1. Пересылка

MOV	A,A	7F
	A,B	78
	A,C	79
	A,D	7A
	A,E	7B
	A,H	7C
	A,L	7D
	A,M	7E

MOV	E,A	5F
	E,B	58
	E,C	59
	E,D	5A
	E,E	5B
	E,H	5C
	E,L	5D
	E,M	5E

MOV	B,A	47
	B,B	40
	B,C	41
	B,D	42
	B,E	43
	B,H	44
	B,L	45
	B,M	46

MOV	H,A	67
	H,B	60
	H,C	61
	H,D	62
	H,E	63
	H,H	64
	H,L	65
	H,M	66

MOV	C,A	4F
	C,B	48
	C,C	49
	C,D	4A
	C,E	4B
	C,H	4C
	C,L	4D
	C,M	4E

MOV	L,A	6F
	L,B	68
	L,C	69
	L,D	6A
	L,E	6B
	L,H	6C
	L,L	6D
	L,M	6E

MOV	D,A	57
	D,B	50
	D,C	51
	D,D	52
	D,E	53
	D,H	54
	D,L	55
	D,M	56

MOV	M,A	77
	M,B	70
	M,C	71
	M,D	72
	M,E	73
	M,H	74
	M,L	75

2. Непосредственная пересылка

MVI	A, байт	3E
	B байт	06
	C байт	OE
	D байт	16
	E, байт	1E
	H байт	26
	L, байт	2E
	M, байт	36
XCHG		EB

3. Непосредственная загрузка

LXI	B, 2 байта	01
	D, 2 байта	11
	H, 2 байта	21
	SP, 2 байта	31

4. Загрузка/хранение

LDAX	B	OA
LDAX	D	1A
LHLD	[адр.]	2A
LDA	[адр.]	3A
STAX	B	02
STAX	D	12
SHLD	[адр.]	22
STA	[адр.]	32

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ КОМАНДЫ

1. Сложение

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86
ADC	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

2. Вычитание

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96
SBB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

3. Увеличение

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34
INX	B	03
	D	13
	H	23
	SP	33

4. Уменьшение

DCR	A	3D
	B	05
	C	OD
	D	15
	E	1D
	H	25
	L	2D
	M	35
DCX	B	OB
	D	1B
	H	2B
	SP	3B

5. Двойное сложение

DAD	B	09
	D	19
	H	29
	SP	39

6. Сдвиг

RLC	07
RRC	OF
RAL	17
RAR	1F

7. Специальные

DAA	27
CMA	2F
STC	37
CMC	3F

8. Непосредственные логические и арифметические

ADI	байт	C6
ACI	байт	CE
SUI	байт	D6
SBI	байт	DE
ANI	байт	E6
XRI	байт	EE
ORI	байт	F6
CPI	байт	FE

9. Логические

ANA	A	A7
	B	AO
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
ORA	A	B7
	B	BO
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
CMP	A	BF
	B	BO
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6

КОМАНДЫ ПЕРЕХОДА

1. Переходы

JMP	[адр.]	C3
JNZ	[адр.]	C2
JZ	[адр.]	CA
JNC	[адр.]	D2
JC	[адр.]	DA
JPO	[адр.]	E2
JPE	[адр.]	EA
JP	[адр.]	F2
JM	[адр.]	FA
PCHL	[адр.]	E9

2. Вызовы

CALL	[адр.]	CD
CNZ	[адр.]	C4
CZ	[адр.]	CC
CNC	[адр.]	D4
CC	[адр.]	DC
CPO	[адр.]	E4
CPE	[адр.]	EC
CP	[адр.]	F4
CM	[адр.]	FC

3. Возврат

RET		C9
RNZ		CO
RZ		C8
RNC		DO
RC		D8
RPO		EO
RPE		E8
RP		FO
RM		F8

4. Рестарт

RST	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	F7
	7	FF

КОМАНДЫ ВВОДА/ВЫВОДА И УПРАВЛЕНИЯ

1. Операции со СТЕКОМ

PUSH	B	C5	POP	B	C1
	D	D5		D	D1
	H	E5		H	E1
	PSW	F5		PSW	F1
	XTHL	E3		SPHL	F9

2. Команды ввода/вывода

OUT	байт	D3
IN	байт	DB

3.Управление

DI	F3
EI	FB
NOP	00
HLT	76

НАБОР КОМАНД ПРОЦЕССОРА КР580ИК80

00	NOP	29	DAD H	53	MOV D,E
01	LXI B, 2 байта	2A	LHLD [адр.]	54	MOV D,H
02	STAX B	2B	DCX H	55	MOV D,L
03	INX B	2C	INR L	56	MOV D,M
04	INR B	2D	DCR L	57	MOV D,A
05	DCR B	2E	MVI L, байт	58	MOV E,B
06	MVI B,байт	2F	CMA	59	MOV E,C
07	RLC	30	-	5A	MOV E,D
08	-	31	LXI SP, 2 байта	5B	MOV E,E
09	DAD B	32	STA [адр.]	5C	MOV E,H
0A	LDAX B	33	INX SP	5D	MOV E,L
0B	DCX B	34	INR M	5E	MOV E,M
0C	INR C	35	DCR M	5F	MOV E,A
0D	DCR C	36	MVI M, байт	60	MOV H,B
0E	MVI C, байт	37	STC	61	MOV H,C
0F	RRC	38	-	62	MOV H,D
		39	DAD SP	63	MOV H,E
10	-	3A	LDA [адр.]	64	MOV H,H
11	LXI D, 2 байта	3B	DCX SP	65	MOV H,L
12	STAX D	3C	INR A	66	MOV H,M
13	INX D	3D	DCR A	67	MOV H,A
14	INR D	3E	MVI A,байт	68	MOV L,B
15	DCR D	3F	CMC	69	MOV L,C
16	MVI D,байт	40	MOV B,B	6A	MOV L,D
17	RAL	41	MOV B,C	6B	MOV L,E
18	-	42	MOV B,D	6C	MOV L,H
19	DAD D	43	MOV B,E	6D	MOV L,L
1A	LDAX D	44	MOV B,H	6E	MOV L,M

1B DCX D
1C INR E
1D DCR E
1E MVI E,байт
1F RAR
20 —
21 LXI H, 2 байта
22 SHLD [адр.]
23 INX H
24 INR H
25 DCR H
26 MVI H, байт
27 DAA
28 -
7D MOV A,L
7E MOV A,M
7F MOV A,A
80 ADD B
81 ADD C
82 ADD D
83 ADD E
84 ADD H
85 ADD L
86 ADD M
87 ADD A
88 ADC B
89 ADC C
8A ADC D
8B ADC E
8C ADC H
8D ADC L
8E ADC M
8F ADC A
90 SUB B
91 SUB C
92 SUB D
93 SUB E
94 SUB H
95 SUB L
96 SUB M
97 SUB A
98 SBB B
99 SBB C
9A SBB D
9B SBB E
9C SBB H
9D SBB L
9E SBB M
9F SBB A
AO ANA B
Al ANA C
A2 ANA D
A3 ANA E
A4 ANA H
A5 ANA L
A6 ANA M
A7 ANA A
A8 XRA B

45 MOV B,L
46 MOV B,M
47 MOV B,A
48 MOV C,B
49 MOV C,C
4A MOV C,D
4B MOV C,E
4C MOV C,H
4D MOV C,L
4E MOV C,M
4F MOV C,A
50 MOV D,B
51 MOV D,C
52 MOV D,D
A9 XRA C
AA XRA D
AB XRA E
AC XRA H
AD XRA L
AE XRA M
AF XRA A
BD ORA B
B1 ORA C
B2 ORA D
B3 ORA E
B4 ORA H
B5 ORA L
B6 ORA M
B7 ORA A
B8 CMP B
B9 CMP C
BA CMP D
BB CMP E
BC CMP H
BD CMP L
BE CMP M
BF CMP A
CO RNZ
C1 POP B
C2 JNZ [адр.]
C3 JMP [адр.]
C4 CNZ [адр.]
C5 PUSH B
C6 ADI байт
C7 RST ø
C8 RZ
C9 RET
CA JZ [адр.]
CB -
CC CZ [адр.]
CD CALL [адр.]
CE ACI байт
CF RST 1
DO RNC
D1 POP D
D2 JNC [адр.]
D3 OUT байт
D4 CNC [адр.]

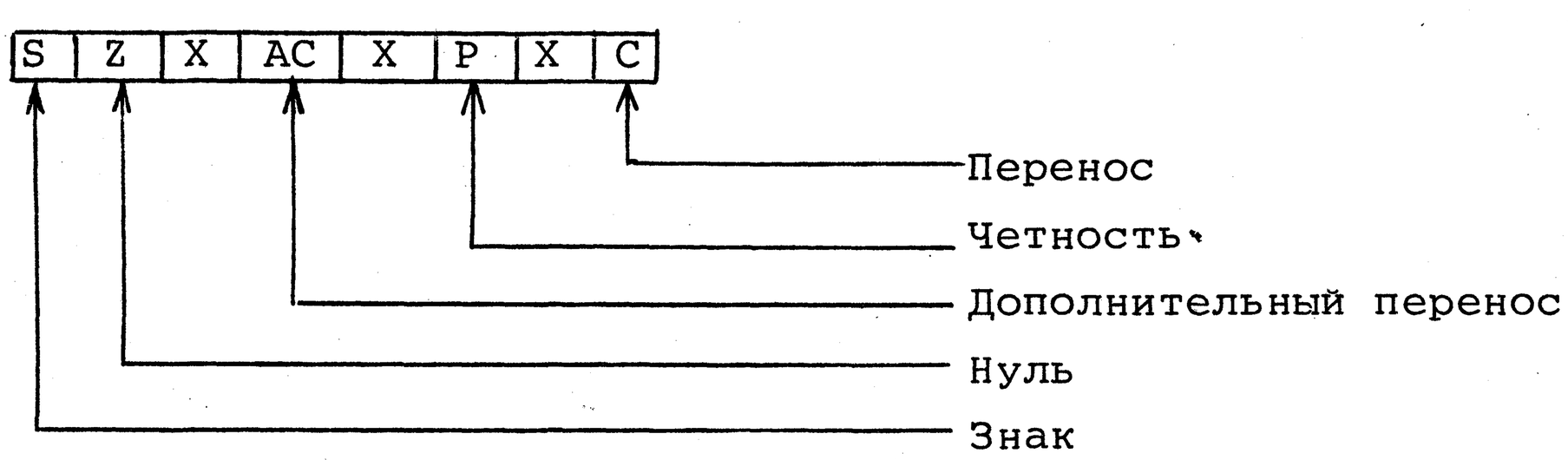
6F MOV L,A
70 MOV M,B
71 MOV M,C
72 MOV M,D
73 MOV M,E
74 MOV M,H
75 MOV M,L
76 HLT
77 MOV M,A
78 MOV A,B
79 MOV A,C
7A MOV A,D
7B MOV A,E
7C MOV A,H
D5 PUSH D
D6 SUI байт
D7 RST 2
D8 RC
D9 -
DA JC [адр.]
DB IN байт
DC CC [адр.]
DD -
DE SBI байт
DF RST 3
EO RPO
E1 POP H
E2 JPO [адр.]
E3 XTHL
E4 CPO [адр.]
E5 PUSH H
E6 ANI байт
E7 RST 4
E8 RPE
E9 PCHL
EA JPE [адр.]
EB XCHG
EC CPE [адр.]
ED -
EE XRI байт
EF RST 5
FO RP
F1 POP PSW
F2 JP [адр.]
F3 DI
F4 CP [адр.]
F5 PUSH PSW
F6 ORI байт
F7 RST 6
F8 RM
F9 SPHL
FA JM [адр.]
FB EI
FC CM [адр.]
FD -
FE CPI байт
FE RST 7

ВНУТРЕННЯЯ ОРГАНИЗАЦИЯ РЕГИСТРОВ

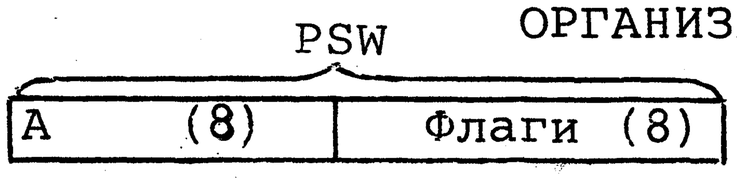
A (8)

B(8)	C(8)
D(8)	E(8)
H(8)	L(8)
Программный счетчик (16)	
Указатель СТЕКА (16)	

D7 Флаговый регистр D0



ОРГАНИЗАЦИЯ РЕГИСТРОВЫХ ПАР



PSW - слово состояния программы

B (B C) (16)
D (D E) (16)
H (H L) (16)
Программный счетчик (16)
Указатель СТЕКА (16)

Примечание. При адресации и арифметических операциях левый байт является старшим. Левый байт первым вводится в СТЕК, правый - первым считывается из СТЕКА.

ОПЕРАЦИИ С ПАРАМИ РЕГИСТРОВ И СТЕКОМ

Команда	PSW (A/F)	Пары регистров B (B/C)	D (D/C)	H (H/L)	SP	PC	Функции
INX		03	13	23	33		Увеличить содержимое пары регистров
DCX		0B	1B	2B	3B		Уменьшить содержимое пары регистров
LDAX		0A	1A	7E(1)			Косвенная загрузка аккумулятора /пара регистров содержит адрес/
STAX		02	12	77(2)			Косвенная запись содержимого аккумулятора в ОЗУ /пара регистров содержит адрес/
LHLD				2A			Прямая загрузка H/L /байты 2 и 3 содержат адрес/

Команда	PSW (A/F)	Пары регистров B (B/C)	D (D/C)	H (H/L)	SP	PC	Функции
SHLD				22			Прямая запись содержимого H/L в ОЗУ /байты 2 и 3 содержат адрес/
LXI		01	11	21	31	03(3)	Непосредственная загрузка пары регистров /байты 2 и 3 содержат непосредственные данные/
PCHL						E9	Загрузка в PC содержимого H/L /переход по адресу, хранимому в H/L/
XGHG			EB				Обмен данными между парами D/E H/L
DAD		09	19	29	39		Сложение содержимого пары регистров с содержимым H/L
PUSH	F5	C5	D5	E5			Записать в СТЕК содержимое пары регистров
POP	F1	C1	D1	E1			Считать из СТЕКА содержимое пары регистров
XTHL				E3			Обмен содержимым между H/L и верхушкой СТЕКА
SPHL					F9		Загрузка SP содержимым H/L

(1) - команда MOV A,M
(2) - команда MOV A,M
(3) - команда JMP

КОМАНДЫ УПРАВЛЕНИЯ ПЕРЕХОДАМИ

Состояние флагов	Переход	Вызов	Возврат
Z=1	JZ CA	CZ CC	RZ C8
Z=0	JNZ C2	CNZ C4	RNZ CO
Перенос есть	JC DA	CC DC	RC D8
Переноса нет	JNC D2	CNC D4	RNC DO
Знак положительный	JP F2	CP F4	RP FO
Знак отрицательный	JMP FA	CM FC	RM F8
Результат четный	JPE EA	CPE EC	RPE E8
Результат нечетный	JPO E2	CPO E4	RPO EO
Безусловное	JMP C3	CALL CD	RET C9

КОМАНДЫ РЕСТАРТА

Команда	Код	Адрес
RST 0	C7	0000_{16}
RST 1	CF	0008_{16}
RST 2	D7	0010_{16}
RST 3	DF	0018_{16}
RST 4	E7	0020_{16}
RST 5	EF	0028_{16}
RST 6	E7	0030_{16}
RST 7	FF	0038_{16}

ОПЕРАЦИИ АККУМУЛЯТОРА

Команда	Код	Функция
XRA A	AF	Очистка аккумулятора и сброс флага переноса
ORA A	B7	Сброс флага переноса
CMC	3F	Инвертирование флага переноса
CMA	2F	Инвертирование содержимого аккумулятора
STC	37	Установка флага переноса
RLC	07	Сдвиг влево
RRC	OF	Сдвиг вправо
RAL	17	Сдвиг влево через флаг переноса
RAR	1F	Сдвиг вправо через флаг переноса
DAA	27	Десятичное дополнение аккумулятора

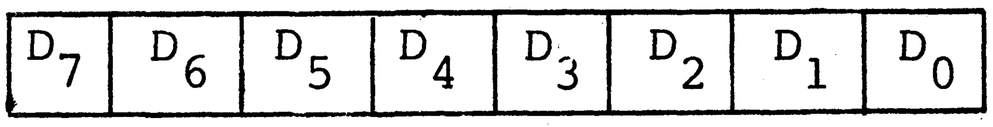
Форматы команд и данных

Память для МП КР580ИК80 организована в 8-битные слова, называемые байтами. Каждый байт имеет уникальный 16-битный двоичный адрес, соответствующий местоположению бай-

та в памяти. КР580ИК80 может адресовать 65, 536 байт памяти, которая состоит из ПЗУ и ОЗУ.

Данные в процессоре записываются в виде 8-битных двоичных чисел

Слово данных

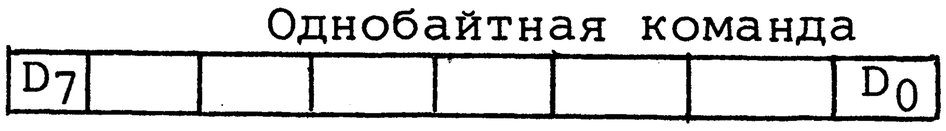


Старший бит Младший бит

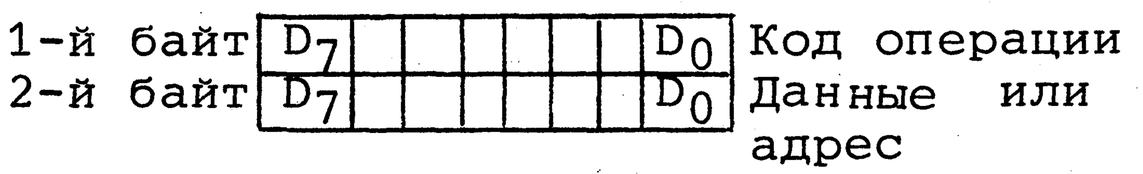
Если регистр или слово данных содержат двоичное число, то необходимо установить порядок, в котором записываются биты числа. В процессоре КР580ИК80 бит 0 обозначается как младший бит, бит 7 - как старший бит.

Команды МП КР580ИК80 могут быть длиной в один, два или три байта. Многобайтные команды должны храниться в соседних ячейках памяти. Адрес первого байта всегда интерпретируется как адрес кода команды. Точный формат команды зависит от конкретной операции, которую надо выполнить.

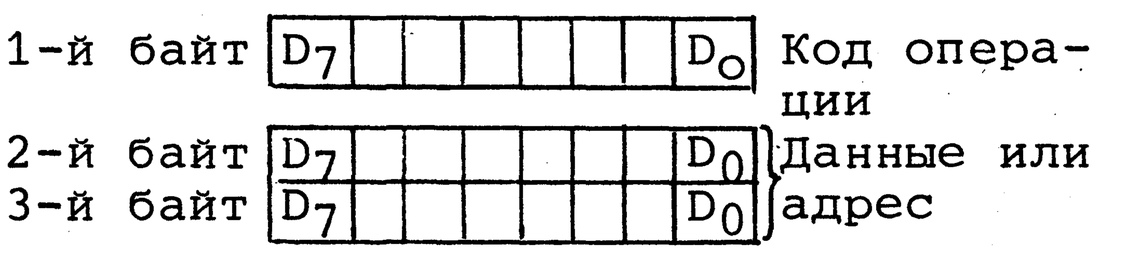
Однобайтная команда



Двухбайтная команда



Трехбайтная команда



Способы адресации

Часто обрабатываемые данные хранятся в памяти. При использовании многобайтных данных такие данные, как команды, хранятся в следующих одна за другой ячейках памяти. Младший байт записывается первым, за ним следуют более старшие байты. МП КР580ИК80 имеет четыре различных способа адресации данных, хранящихся в памяти или в регистрах.

Прямая - второй и третий байты команды содержат точный адрес ячейки памяти, в которой хранятся нужные данные. Младшие биты адреса содержатся во втором байте, старшие - в третьем.

Регистровая - команда указывает регистр или пару регистров, где содержатся данные.

Косвенная регистровая - команда указывает пару регистров, в которой содержится адрес ячейки памяти, где расположены нужные данные. Старшие биты адреса хранятся в первом регистре пары, младшие - во втором

Непосредственная - команда сама содержит нужные данные. Данные могут быть длиной 8 или 16 бит /сначала идет младший байт, затем старший/.

Если нет команд прерывания или перехода, процессор выполняет команды, хранящиеся в ячейках памяти, следующих по порядку одна за другой. Команда перехода может определить адрес следующей команды, которую нужно исполнить, одним из двух способов.

Прямой - команда перехода содержит адрес следующей команды, которую нужно исполнить. За исключением команды RST второй байт содержит младшие биты адреса, третий байт - старшие биты.

Косвенный регистровый - команда перехода указывает пару регистров, в которой находится адрес следующей исполняемой команды. Старшие биты адреса хранятся в первом регистре пары, младшие - во втором.

Команда RST - специальная однобайтная команда вызова /обычно используется при прерываниях/, содержит трехбитное поле. Управление программой передается команде, адрес которой равен содержимому трехбитного поля, умноженному на 8.

ФЛАГИ

В МП КР580ИК80 имеются пять условных флагов, связанных с исполнением команд: нуля Z, знака S, четности P, переноса CY, вспомогательного переноса AC. Каждый флаг представляет собой однобитный регистр. Флаг установлен, если бит равен

логической 1, флаг сброшен, если бит равен логическому 0.

Флаг нуля (Z) - устанавливается, если результат операции равен 0, сбрасывается, если результат не равен 0.

Флаг знака (S) - устанавливается, если старший бит результата операции равен 1, в противном случае флаг сбрасывается.

Флаг четности (P) - если сумма по модулю 2 бит в результате операции равна 0 /т.е. результат содержит четное число единицы/, то флаг устанавливается. Если результат содержит нечетное число единиц, флаг сбрасывается.

Флаг переноса (CY) - если при выполнении операции происходит перенос /при сложении/ или заем из более старшего бита /при вычитании и сравнении/, то флаг устанавливается, в противном случае - сбрасывается.

Вспомогательный перенос (AC) - если при выполнении операции происходит перенос из 3-го в 4-й бит результата, то флаг устанавливается. Если этого не происходит, флаг сбрасывается. Этот флаг устанавливается при сложении, вычитании, уменьшении, увеличении, сравнении и логических операциях, но обычно он используется при сложении и сравнении, предшествующих команде DAA /десятичное дополнение аккумулятора/.

СИМВОЛЫ И СОКРАЩЕНИЯ

Аккумулятор	A - регистр
адр.	16-битный адрес
Байт	8-битное слово данных
2 байта	18-битное слово данных
Байт 2	Второй байт команды
Байт 3	Третий байт команды
Порт	8-битный адрес устройства ввода/вывода
r, r1, r2	Один из регистров A, B, C, D, E, H, L
DDD, SSS	Набор бит, означающий один из регистров A, B, C, D, E, H, L (DDD - регистр назначения, SSS - регистр источника)

DDD (SSS)	Регистр
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp	Одна из регистровых пар: B - пара B и C (B - старший регистр, C - младший регистр); D - представляет пару D и E (D - старший регистр, E - младший регистр); H - пара H и L (H - старший регистр, L - младший регистр)
SP	16-битный регистр указателя СТЕКА

Набор битов, определяющий пару регистров:

RP	Пара регистров
00	B-C
01	D-E
10	H-L
11	SP

rh	Первый (старший) регистр указанной пары
rl	Второй (младший) регистр указанной пары
PC	16-битный регистр программного счетчика (обозначения PCH и PCL используется для указания старших и младших 8 битов соответственно)
SP	16-битный регистр указателя СТЕКА (обозначения SPH и SPL используются для указания старших и младших 8 битов соответственно)
r_m	Бит m регистра r (биты нумеруются от 7 до 0 слева направо)
Z	Флаг нуля
S	Флаг знака
P	Флаг четности
CY	Флаг переноса
AC	Флаг вспомогательного переноса
( )	В скобки заключается содержимое ячейки памяти или регистра
←	"Пересылается в"
∧	Логическая "И"
⊻	"Исключающее ИЛИ"
∨	Логическое "ИЛИ"
+	Сложение
-	Вычитание с дополнением до 2
x	Умножение
↔	"Обменивается с"
—	Дополнение до 1, т.е. инверсия
n	Номер рестарта от 0 до 7
NNN	Двоичное представление кода рестарта от 000 до 111 /от 0 до 7/

КОМАНДЫ ПЕРЕСЫЛКИ

Команды этой группы осуществляют пересылку данных в/из регистр(а) и ячейки памяти и не изменяют состояния флагов.

MOV r1, r2 (ПЕРЕСЫЛКА ДАННЫХ ИЗ РЕГИСТРА В РЕГИСТР)

(r1) ← (r2)

Содержимое регистра r2 пересылается в регистр r1

0	1	D	D	D	S	S	S

MOV A,A	7F	MOV B,A	47	MOV C,B	48	MOV D,B	50
MOV A,B	78	MOV B,B	40	MOV C,C	49	MOV D,C	51
MOV A,C	79	MOV B,C	41	MOV C,D	4A	MOV D,D	52
MOV A,D	7A	MOV B,D	42	MOV C,E	4B	MOV D,E	53
MOV A,E	7B	MOV B,E	43	MOV C,H	4C	MOV D,H	54
MOV A,H	7C	MOV B,H	44	MOV C,L	4D	MOV D,L	55
MOV A,L	7D	MOV B,L	45				
MOV C,A	4F	MOV D,A	57	MOV E,A	5F	MOV H,A	67
				MOV E,B	58	MOV H,B	60
				MOV E,C	59	MOV H,E	61
				MOV E,D	5A	MOV H,D	62
				MOV E,E	5B	MOV H,E	63
				MOV E,H	5C	MOV H,H	64
				MOV E,L	5D	MOV H,L	65
				MOV L,A			
				MOV L,B			
				MOV L,C			
				MOV L,D			
				MOV L,E			
				MOV L,H			
				MOV L,L			

Циклы: 1

Состояния: 4
Адресация: регистровая
Флаги: не используются

MOV r, M (ПЕРЕСЫЛКА ДАННЫХ ИЗ ПАМЯТИ)
(r) ⟵ ((H) (L))

Содержимое ячейки памяти, адрес которой хранится в регистрах H и L, пересылается в регистр r.

0	1	D	D	D	1	1	0

MOV A,M 7E
MOV B,M 46
MOV C,M 4E
MOV D,M 56
MOV E,M 5E
MOV H,M 66
MOV L,M 6E

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: нет

MOV M, r (ПЕРЕСЫЛКА ДАННЫХ В ПАМЯТЬ)
((H) (L)) ⟵ (r)

Содержимое регистра r пересылается в ячейку памяти, адрес которой содержится в регистрах H и L.

0	1	1	1	0	S	S	S

MOV M,A 77
MOV M,B 70
MOV M,C 71
MOV M,D 72
MOV M,E 73
MOV M,H 74
MOV M,L 75

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: нет

MVI, r байт (НЕПОСРЕДСТВЕННАЯ ПЕРЕСЫЛКА)
(r) ⟵ (байт 2)

Содержимое второго байта команды пересылается в регистр n

0	0	D	D	D	1	1	0
Данные							

MVI A, байт 3E
MVI B, байт 06
MVI C, байт OE
MVI D, байт 16
MVI E, байт 1E
MVI H, байт 26
MVI L, байт 2E

Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: нет

MVI M, байт (НЕПОСРЕДСТВЕННАЯ ПЕРЕСЫЛКА В ПАМЯТЬ)

((H) (L)) ⟵ (байт 2)

Содержимое второго байта команды пересылается в ячейку памяти, адрес которой указан в регистрах H и L.

0	0	1	1	0	1	1	0
Данные							

MVI M, байт 36

Циклы: 3
Состояния: 10
Адресация: непосредственная/косвенная регистровая
Флаги: нет

LXI rp, 2 байта (НЕПОСРЕДСТВЕННАЯ ЗАГРУЗКА ПАРЫ РЕГИСТРОВ)

(rh) ⟵ (байт 3)
(rl) ⟵ (байт 2)

Третий байт команды пересылается в старший регистр (rh) пары регистров, второй - в младший регистр (rl) пары регистров.

0	0	R	P	0	0	0	1
Младший байт данных							
Старший байт данных							

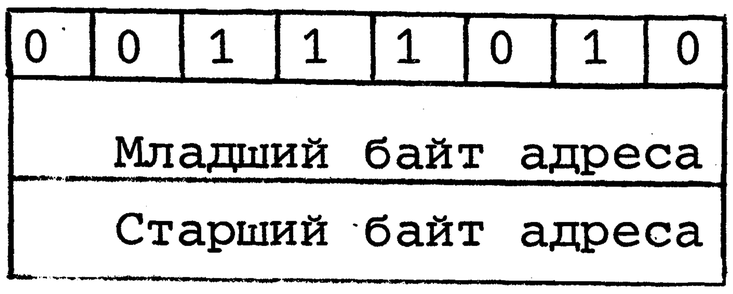
LXI B, 2 байта 01 (непосредственная загрузка пары регистров B и C)
LXI D, 2 байта 11 (непосредственная загрузка данных пары регистров D и E)

LXI H, 2 байта 21 (непосредственная загрузка пары регистров H и L)
LXI SP, 2 байта 31 (непосредственная загрузка указателя СТЕКА)

Циклы: 3
Состояния: 10
Адресация: непосредственная
Флаги: нет

LDA [адр.] (ПРЯМАЯ ЗАГРУЗКА АККУМУЛЯТОРА)
(A) ← ((байт 3) (байт 2))

Содержимое ячейки памяти, адрес которой указан во втором и третьем байтах команды, загружается в аккумулятор.

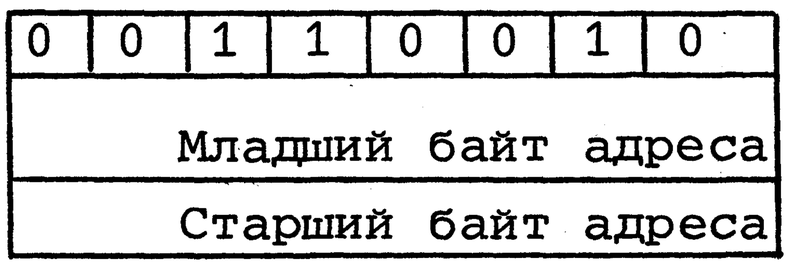


LDA [адр.] 3A
Циклы: 4
Состояния: 13
Адресация: прямая
Флаги: нет

STA [адр.] (ПРЯМАЯ ЗАПИСЬ СОДЕРЖИМОГО АККУМУЛЯТОРА В ПАМЯТЬ)

((байт 3) (байт 2)) ← (A)

Содержимое аккумулятора пересылается в ячейку памяти, адрес которой указан во втором и третьем байтах команды.

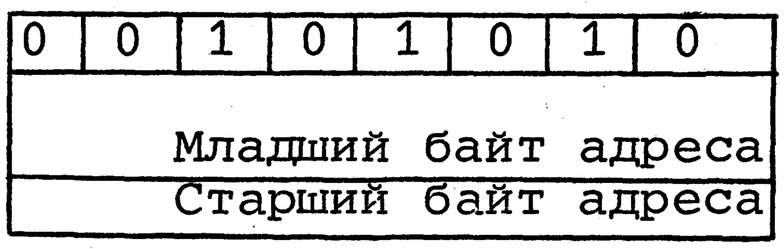


STA [адр.] 32
Циклы: 4
Состояния: 13
Адресация: прямая
Флаги: нет

LHLD [адр.] (ПРЯМАЯ ЗАГРУЗКА H и L)

(L) ← ((байт 3) (байт 2))
(H) ← ((байт 3) (байт 2) + 1)

Содержимое ячейки памяти, адрес которой указан во втором и третьем байтах команды, пересылается в регистр L. Содержимое ячейки памяти со следующим адресом ((адр.) + 1) пересылается в регистр H.

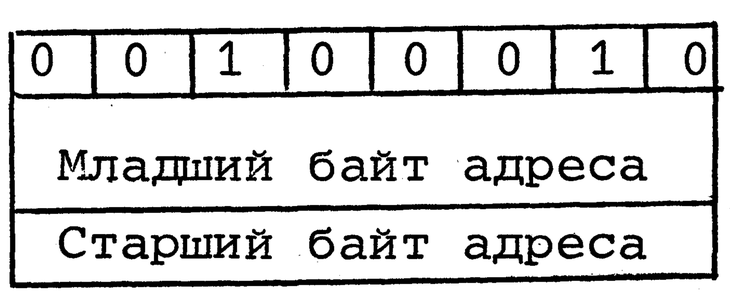


LHLD [адр.] 2A
Циклы: 5
Состояния: 16
Адресация: прямая
Флаги: нет

SHLD [адр.] (ПРЯМАЯ ЗАПИСЬ СОДЕРЖИМОГО H, L РЕГИСТРОВ В ПАМЯТЬ)

((байт 3) (байт 2)) ← (L)
((байт 3) (байт 2) + 1) ← (H)

Содержимое регистра L пересылается в ячейку памяти, адрес которой определен во втором и третьем байтах команды. Содержимое H регистра пересылается в следующую ячейку памяти.



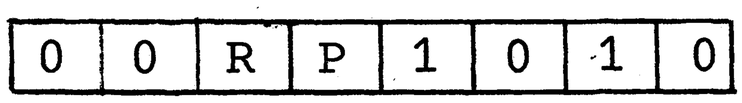
SHLD [адр.] 22
Циклы: 5
Состояния: 16
Адресация: прямая
Флаги: нет

LDAX rp (КОСВЕННАЯ ЗАГРУЗКА АККУМУЛЯТОРА)

(A) ← ((rp))

Содержимое ячейки памяти, адрес которой указан в паре регистров rp, пересылается в A-регистр.

Примечание. Могут быть указаны только пары регистров rp=B /регистры B и C/ и rp=D /регистры D и E/.



LDAX B 0A

LDAX D 1A

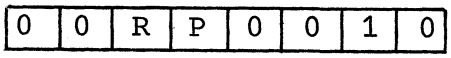
Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: нет

STAX rp (КОСВЕННАЯ ЗАПИСЬ СОДЕРЖИМОГО АККУМУЛЯТОРА В ПАМЯТЬ)

((rp)) ⟵ (A)

Содержимое аккумулятора пересылается в ячейку памяти, адрес которой определен в паре регистров rp.

Примечание. Могут быть указаны только пары регистров rp=B /регистры B и C/ или rp=D /регистры D и E/.



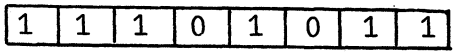
STAX B 02
STAX D 12

Циклы: 2
Состояния: 7
Адресация: косвенная регистрация
Флаги: нет

XCHG (ОБМЕН ДАННЫМИ МЕЖДУ РЕГИСТРАМИ H, L и D, E)

(H) ⟷ (D)
(L) ⟷ (E)

Содержимое регистров H и L обменивается с содержимым регистров D и E.



XCHG EB

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: нет

АРИФМЕТИЧЕСКИЕ КОМАНДЫ

Команды этой группы выполняют арифметические операции над данными в регистрах и ячейках памяти.

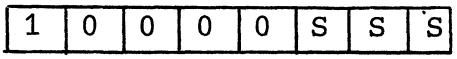
Если нет особых указаний, то команды изменяют состояния флагов нуля, четности, переноса и вспомогательного переноса в соответствии со стандартными правилами.

Все операции вычитания выполняются через арифметическое дополнение до 2, причем флаг переноса устанавливается в логическую 1, если имеется заем, и сбрасывается, если заема не было.

ADD r (СЛОЖЕНИЕ СОДЕРЖИМОГО АККУМУЛЯТОРА С СОДЕРЖИМЫМ РЕГИСТРА)

(A) ⟵ (A)+(r)

Содержимое регистра r складывается с содержимым аккумулятора. Результат помещается в аккумулятор.



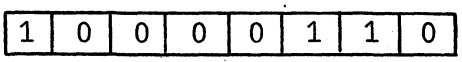
ADD A 87
ADD B 80
ADD C 81
ADD D 82
ADD E 83
ADD H 84
ADD L 85

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

ADD M (СЛОЖЕНИЕ СОДЕРЖИМОГО АККУМУЛЯТОРА С СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ)

(A) ⟵ (A) + ((H) (L))

Содержимое ячейки памяти, адрес которой содержится в регистрах H и L, складывается с содержимым аккумулятора. Результат помещается в аккумулятор.



ADD M 86

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

ADI байт (НЕПОСРЕДСТВЕННОЕ СЛОЖЕНИЕ)

(A) ⟵ (A) + (байт 2)

Содержимое второго байта команды складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1	1	0	0	0	1	1	0
Данные							

ADI байт C6
Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

ADC r (СЛОЖЕНИЕ СОДЕРЖИМОГО АККУМУЛЯТОРА С СОДЕРЖИМЫМ РЕГИСТРА И БИТОМ ФЛАГА ПЕРЕНОСА)

(A) ← (A)+(r)+(CY)

Содержимое регистра r и бит флага переноса складываются с содержимым аккумулятора. Результат помещается в аккумулятор.

1	0	0	0	1	S	S	S

ADC A 8F
ADC B 88
ADC C 89
ADC D 8A
ADC E 8B
ADC H 8C
ADC L 8D

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

ADC M (СЛОЖЕНИЕ СОДЕРЖИМОГО АККУМУЛЯТОРА С СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ И БИТОМ ПЕРЕНОСА)

(A) ← (A)+((H)(L))+(CY)

Содержимое ячейки памяти, адрес которой указан в регистрах H и L, и содержимое бита флага переноса складываются с содержимым аккумулятора. Результат помещается в аккумулятор.

1	0	0	0	1	1	1	0

ADC M 8E
Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

ACI байт (НЕПОСРЕДСТВЕННОЕ СЛОЖЕНИЕ С БИТОМ ПЕРЕНОСА)

(A) ← (A)+(байт 2) + (CY)

Содержимое второго байта команды и бита флага переноса складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1	1	0	0	1	1	1	0
Данные							

ACI байт CE
Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

SUB r (ВЫЧИТАНИЕ СОДЕРЖИМОГО РЕГИСТРА)

(A) ← (A)-(r)

Содержимое регистра r вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

1	0	0	1	0	S	S	S

SUB A 97
SUB B 90
SUB C 91
SUB D 92
SUB E 93
SUB H 94
SUB L 95

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

SUB M (ВЫЧИТАНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ)

(A) ← (A)-((H)(L))

Содержимое ячейки памяти, адрес которой содержится в H и L регистрах, вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

1	0	0	1	0	1	1	0

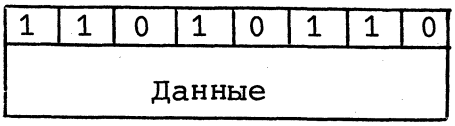
SUB M 96
Циклы: 2
Состояния: 7

Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

SUI байт (НЕПОСРЕДСТВЕННОЕ ВЫЧИТАНИЕ)

(A) ← (A)-(байт 2)

Содержимое второго байта команды вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

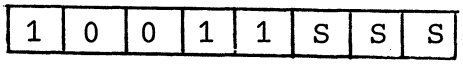


SUI байт D6
Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

SBB r (ВЫЧИТАНИЕ СОДЕРЖИМОГО РЕГИСТРА И БИТА ПЕРЕНОСА)

(A) ← (A)-(r)-(CY)

Содержимое регистра r и бит флага переноса CY вычитаются из содержимого аккумулятора. Результат помещается в аккумулятор.



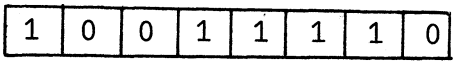
SBB A 9F
SBB B 98
SBB C 99
SBB D 9A
SBB E 9B
SBB H 9C
SBB L 9D

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

SBB M (ВЫЧИТАНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ И БИТА ПЕРЕНОСА)

(A) ← (A)-((H)(L))-(CY)

Содержимое ячейки памяти, адрес которой содержится в регистрах H и L, и содержимое бита флага переноса CY вычитаются из содержимого аккумулятора. Результат помещается в аккумулятор.

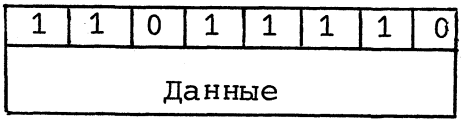


SBB M 9E
Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

SBI байт (НЕПОСРЕДСТВЕННОЕ ВЫЧИТАНИЕ С ЗАЕМОМ)

(A) ← (A)-(байт 2) - (CY)

Содержимое второго байта команды и бита флага переноса CY вычитаются из аккумулятора. Результат помещается в аккумулятор.



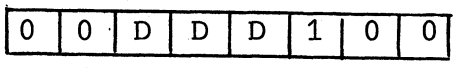
SBI байт DE

Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

INR r (УВЕЛИЧЕНИЕ СОДЕРЖИМОГО РЕГИСТРА)
(r) ← (r)+1

Содержимое регистра r увеличивается на единицу.

Примечание. Команда влияет на состояние всех флагов, кроме флага переноса CY.



INR A 3C
INR B 04
INR C 0C
INR D 14
INR E 1C
INR H 24
INR L 26

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, AC

INR M (УВЕЛИЧЕНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ)

((H)(L)) ← ((H)(L))+1

Содержимое ячейки памяти, адрес которой содержится в регистрах H и L, увеличивается на единицу.

Примечание. Команда воздействует на все флаги, кроме флага переноса CY.

0	0	1	1	0	1	0	0

INR M 34
Циклы: 3
Состояния: 10
Адресация: косвенная регистровая
Флаги: Z, S, P, AC

DCR r (УМЕНЬШЕНИЕ СОДЕРЖИМОГО РЕГИСТРА)
(r) ← (r)-1

Содержимое регистра r уменьшается на единицу.

Примечание. Команда воздействует на все флаги, кроме флага переноса CY.

0	0	D	D	D	1	0	1

DCR A 3D
DCR B 05
DCR C 0D
DCR D 15
DCR E 1D
DCR H 25
DCR L 2D

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, AC

DCR M (УМЕНЬШЕНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ)

((H) (L)) ← ((H)(L))-1

Содержимое ячейки памяти, адрес которой содержится в регистрах H и L, уменьшается на единицу.

Примечание. Команда воздействует на все флаги, кроме флага переноса CY.

0	0	1	1	0	1	0	1

DCR M 35
Циклы: 3
Состояния: 10
Адресация: косвенная регистровая
Флаги: Z, S, P, AC

INX rp (УВЕЛИЧЕНИЕ СОДЕРЖИМОГО ПАРЫ РЕГИСТРОВ)

(rh) (rl) ← (rh)(rl)+1

Содержимое пары регистров rp увеличивается на единицу.

Примечание. Команда не воздействует на флаги.

0	0	R	P	0	0	1	1

INX B 03 (пара регистров B и C)
INX D 13 (пара регистров D и E)
INX H 23 (пара регистров H и L)
INX SP 33 (указатель СТЕКА)

Циклы: 1
Состояния: 6
Адресация: регистровая
Флаги: нет

DCX rp (УМЕНЬШЕНИЕ СОДЕРЖИМОГО ПАРЫ РЕГИСТРОВ)

(rh)(rl) ← (rh)(rl)-1

Содержимое пары регистров rp уменьшается на единицу.

Примечание. Команда не воздействует на флаги.

0	0	R	P	1	0	1	1

DCX B 0B (пара регистров B и C)
DCX D 1B (пара регистров D и E)
DCX H 2B (пара регистров H и L)
DCX SP 3B (указатель СТЕКА)

Циклы: 1
Состояния: 6
Адресация: регистровая
Флаги: нет

DAD rp (СЛОЖЕНИЕ СОДЕРЖИМОГО ПАРЫ РЕГИСТРОВ С СОДЕРЖИМЫМ РЕГИСТРОВ H и L)
(H)(L) ← (H)(L)+(rh)(rl)

Содержимое пары регистров rp складывается с содержимым регист-

ров H и L. Результат помещается в пару регистров H, L.

Примечание. Используется только флаг переноса CY. Он устанавливается, если имеется перенос при сложении с двойной точностью, в противном случае он сбрасывается.

0	0	R	P	1	0	0	1

DAD B 09 (пара регистров B и C)
DAD D 19 (пара регистров D и E)
DAD H 29 (пара регистров H и L)
DAD SP 39 (указатель СТЕКА)

Циклы: 3
Состояния: 10
Адресация: регистровая
Флаги: CY

DAA (ДЕСЯТИЧНОЕ ДОПОЛНЕНИЕ АККУМУЛЯТОРА)

8-битное число в аккумуляторе дополнятеся до представления в виде двух 4-битных чисел в двоично-десятичном коде с помощью следующих операций.

1. Если число, представленное четырьмя младшими битами аккумулятора, больше 8 или установлен флаг вспомогательного переноса AC, то к содержимому аккумулятора добавляется 6.

2. Если теперь число, представленное четырьмя старшими битами аккумулятора, больше 9 или установлен флаг переноса CY, то число 6 добавляется к числу, образованному четырьмя старшими битами аккумулятора.

Примечание. Используются все флаги.

0	0	1	0	0	1	1	1

DAA 27

Циклы: 1
Состояния: 4
Флаги: Z, S, P, CY, AC

ЛОГИЧЕСКИЕ КОМАНДЫ

Команды этой группы выполняют логические /булевы/ операции над данными, хранящимися в регистрах и ячейках памяти и флагами.

Если нет особых указаний, то все команды воздействуют на флаги: нуля, четности, знака, переноса и вспомогательного переноса в соответствии со стандартными правилами.

ANA r (ОПЕРАЦИЯ "И" НАД СОДЕРЖИМЫМ РЕГИСТРА И АККУМУЛЯТОРА)

(A) ← (A) ∧ (r)

Над содержимым регистра r и аккумулятора выполняется операция логическое "И". Результат помещается в аккумулятор. Сбрасывается флаг переноса CY, устанавливается флаг вспомогательного переноса AC.

1	0	1	0	0	S	S	S

ANA A A7
ANA B A0
ANA C A1
ANA D A2
ANA E A3
ANA H A4
ANA L A5

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, CY, AC, P

ANA M (ОПЕРАЦИЯ "И" НАД СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ И АККУМУЛЯТОРА)

(A) ← (A) ∧ ((H)(L))

Над содержимым ячейки памяти, адрес которой содержится в регистрах H и L, и содержимым аккумулятора выполняется операция логическая "И". Результат помещается в аккумулятор. Сбрасывается флаг переноса CY и вспомогательного переноса AC.

1	0	1	0	0	1	1	0

ANA M A6
Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, AC, CY

ANI байт (НЕПОСРЕДСТВЕННАЯ ОПЕРАЦИЯ "И")

(A) ← (A) ∧ (байт 2)

Над содержимым второго байта команды и аккумулятора выполняется операция логическая "И". Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	1	1	0	0	1	1	0
Данные							

ANI байт E6
Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

XRA r (ОПЕРАЦИЯ "ИСКЛЮЧАЮЩЕЕ ИЛИ" НАД СОДЕРЖИМЫМ АККУМУЛЯТОРА И РЕГИСТРА)

(A) ← (A) ∀ (r)

Над содержимым регистра r и аккумулятора выполняется операция "исключающее ИЛИ". Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	0	1	0	1	S	S	S

XRA A AF
XRA B A8
XRA C A9
XRA D AA
XRA E AB
XRA H AC
XRA L AD

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

XRA M (ОПЕРАЦИЯ "ИСКЛЮЧАЮЩЕЕ ИЛИ" НАД СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ И АККУМУЛЯТОРА)

(A) ← (A) ∀ ((H) (L))

Над содержимым ячейки памяти, адрес которой содержится в регистрах H и L, и содержимым аккумулятора выполняется операция "Исключающее ИЛИ". Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	0	1	0	1	1	1	0

XRA M AE

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

XRI байт (НЕПОСРЕДСТВЕННАЯ ОПЕРАЦИЯ "ИСКЛЮЧАЮЩЕЕ ИЛИ")

(A) ← (A) ∀ (байт 2)

Над содержимым аккумулятора и второго байта команды выполняется операция "исключающее ИЛИ". Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	1	1	0	1	1	1	0
Данные							

XRI байт EE
Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

ORA r (ЛОГИЧЕСКОЕ СЛОЖЕНИЕ ("ИЛИ") АККУМУЛЯТОРА С СОДЕРЖИМЫМ РЕГИСТРА)

(A) ← (A) ∨ (r)

Содержимое регистра r логически складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	0	1	1	0	S	S	S

ORA A B7
ORA B B0
ORA C B1
ORA D B2
ORA E B3
ORA H B4
ORA L B5

Циклы: 1

Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

ORA M (ЛОГИЧЕСКОЕ СЛОЖЕНИЕ ("ИЛИ") СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ ИЛИ АККУМУЛЯТОРА)

(A) ← (A) ∨ ((H)(L))

Содержимое ячейки памяти, адрес которой определен в регистрах H и L, логически складывается с содержимым аккумулятора. Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	0	1	1	0	1	1	0

ORA M B6

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

ORI байт (НЕПОСРЕДСТВЕННОЕ ЛОГИЧЕСКОЕ СЛОЖЕНИЕ)

(A) ← (A) ∨ (байт 2)

Содержимое второго байта команды складывается непосредственно с содержимым аккумулятора. Результат помещается в аккумулятор. Сбрасываются флаги переноса CY и вспомогательного переноса AC.

1	1	1	1	0	1	1	0
Данные							

ORI байт F6

Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

CMP r (СРАВНЕНИЕ СОДЕРЖИМОГО РЕГИСТРА С СОДЕРЖИМЫМ АККУМУЛЯТОРА)

(A) - (r)

Содержимое регистра r вычитается из содержимого аккумулятора. Аккумулятор не изменяется. По результату вычитания флаги устанавливаются следующим образом: флаг нуля Z - в 1, если (A) = (r); флаг переноса CY - в 1, если (A) < (r).

1	0	1	1	1	S	S	S

CMP A BF
CMP B B8
CMP C B9
CMP D BA
CMP E BB
CMP H BC
CMP L BD

Циклы: 1
Состояния: 4
Адресация: регистровая
Флаги: Z, S, P, CY, AC

CMP M (СРАВНЕНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ С СОДЕРЖИМЫМ АККУМУЛЯТОРА)

(A) - ((H) (L))

Содержимое ячейки памяти, адрес которой определен в регистрах H и L, вычитается из содержимого аккумулятора. Аккумулятор не изменяется. Флаги устанавливаются по результату вычитания следующим образом: флаг нуля Z - в 1, если (A) = ((H) (L)); флаг переноса CY - в 1, если A < ((H) (L)).

1	0	1	1	1	1	1	0

CMP M BE

Циклы: 2
Состояния: 7
Адресация: косвенная регистровая
Флаги: Z, P, S, CY, AC

CPI (байт) (НЕПОСРЕДСТВЕННОЕ СРАВНЕНИЕ)

(A) - (байт 2)

Содержимое второго байта команды вычитается из содержимого аккумулятора. По результату вычитания флаги устанавливаются следующим образом: флаг нуля Z - в 1, если (A) = (байт 2); флаг переноса CY - в 1, если (A) < (байт 2).

1	1	1	1	1	1	0
Данные						

CPI байт FE

Циклы: 2
Состояния: 7
Адресация: непосредственная
Флаги: Z, S, P, CY, AC

RLC (ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО)

$(A_{n+1}) \leftarrow (A_n)$, $(A_0) \leftarrow (A_7)$, $(CY) \leftarrow (A_7)$

Содержимое аккумулятора сдвигается влево на одну позицию. Содержимое самого старшего бита заносится в младший бит и бит флага переноса. Команда воздействует только на флаг переноса CY.

0	0	0	0	0	1	1	1

RLG 07

Циклы: 1
Состояния: 4
Флаги: CY

RRC(ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО)

$(A_n) \leftarrow (A_{n+1})$, $(A_7) \leftarrow (A_0)$, $(CY) \leftarrow (A_0)$

Содержимое аккумулятора сдвигается на одну позицию вправо. Содержимое младшего бита заносится в самый старший бит и бит флага переноса. Команда воздействует только на флаг переноса CY.

0	0	0	0	1	1	1	1

RRC OF

Циклы: 1
Состояния: 4
Флаги: CY

RAL /СДВИГ ВЛЕВО ЧЕРЕЗ ПЕРЕНОС/

$(A_{n+1}) \leftarrow (A_n)$, $(CY) \leftarrow (A_7)$, $(A_0) \leftarrow (CY)$

Содержимое аккумулятора сдвигается на одну позицию влево через бит флага переноса CY. Младший бит устанавливается равным флагу переноса, а бит флага переноса CY - равным величине старшего бита аккумулятора. Команда воздействует только на флаг переноса CY.

0	0	0	1	0	1	1	1

RAL 17

Циклы: 1
Состояния: 4
Флаги: CY

RAR (СДВИГ ВПРАВО ЧЕРЕЗ ПЕРЕНОС)

$(A_n) \leftarrow (A_{n+1})$, $(CY) \leftarrow (A_0)$, $(A_7) \leftarrow (CY)$

Содержимое аккумулятора сдвигается на одну позицию вправо через бит флага переноса. Старший бит аккумулятора устанавливается равным флагу переноса, флаг переноса - равным младшему биту аккумулятора. Используется только флаг переноса CY.

0	0	0	1	1	1	1	1

RAR 1F

Циклы: 1
Состояния: 4
Флаги: CY

CMA (ДОПОЛНЕНИЕ АККУМУЛЯТОРА)

$(A) \leftarrow (\overline{A})$

Содержимое аккумулятора инвертируется (бит, равный 1, становится равным 0; бит, равный 0, становится равным 1). Флаги не используются.

0	0	1	0	1	1	1	1

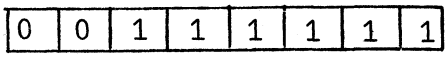
CMA 2F

Циклы: 1
Состояния: 4
Флаги: нет

CMC (ДОПОЛНЕНИЕ ФЛАГА ПЕРЕНОСА)

$CY \leftarrow (\overline{CY})$

Инвертируется бит флага переноса CY. Другие флаги не используются.



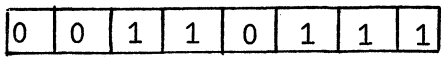
0	0	1	1	1	1	1	1

CMC 3F

Циклы: 1
Состояния: 4
Флаги: CY

STC (УСТАНОВКА ФЛАГА ПЕРЕНОСА)
(CY) ← 1

Бит флага переноса устанавливается в 1. Другие флаги не используются.



0	0	1	1	0	1	1	1

STC 37

Циклы: 1
Состояния: 4
Флаги: CY

КОМАНДЫ ПЕРЕХОДА

Команды этой группы изменяют нормальное последовательное исполнение программы. Исполнение команд не влияет на состояние флагов.

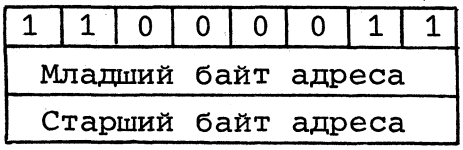
Имеется два типа команд перехода: безусловные и условные. Безусловные команды перехода осуществляют передачу управления, изменяя значение программного счетчика PC. Команды условного перехода сначала проверяют состояние одного из четырех флагов процессора, чтобы определить, выполнено ли указанное в команде условие. Имеются следующие условия перехода:

Условие	Код
NZ - не ноль (Z=0)	000
Z - ноль (Z=1)	001
NC - нет переноса (CY=0)	010
C - есть перенос (CY=1)	011
PO - нечетный результат (P=0)	100
PE - четный результат (P=1)	101
P - плюс (S=0)	110
M - минус (S=1)	111

JMP [адр.] (ПЕРЕХОД)

(PC) ← (байт 3) (байт 2)

Управление передается команде, адрес которой указан во втором и третьем байтах команды перехода.



1	1	0	0	0	0	1	1
Младший байт адреса							
Старший байт адреса							

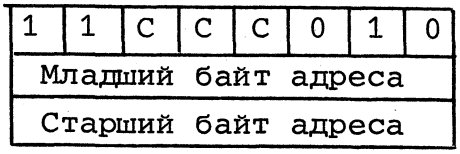
JMP [адр.] C3

Циклы: 3
Состояния: 10
Адресация: непосредственная
Флаги: нет

J (УСЛОВИЕ) [адр.] (УСЛОВНЫЙ ПЕРЕХОД)

Если (CCC), то (PC) ← (байт 3) (байт 2)

Если указанное условие истинно, то управление передается команде, адрес которой указан во втором и третьем байтах команды перехода. Если условие ложное, то последовательный код программы не изменяется.



1	1	C	C	C	0	1	0
Младший байт адреса							
Старший байт адреса							

JNZ	[адр.]	C2
JZ	[адр.]	CA
JNC	[адр.]	D2
JC	[адр.]	DA
JPO	[адр.]	E2
JPE	[адр.]	EA
JP	[адр.]	F2
JM	[адр.]	FA

Циклы: 3
Состояния: 10
Адресация: непосредственная
Флаги: нет

CALL [адр.] (ВЫЗОВ)

((SP)-1) ← (PCH)
((SP)-2) ← (PCL)
(SP) ← (SP)-2
(PC) ← (байт 3) (байт 2)

Старшие 8 бит адреса следующей команды пересылаются в ячейку памяти, адрес которой на 1 меньше содержимого указателя СТЕКА SP. Младшие 8 бит адреса следующей команды пересылаются в ячейку памяти, адрес которой на 2 меньше величины указателя СТЕКА SP. Содержимое указателя СТЕКА уменьшается на 2. Управление передается команде, адрес которой указан во втором и третьем байтах команды вызова.

1	1	0	0	1	1	0	1
Младший байт адреса							
Старший байт адреса							

CALL [адр.] CD

Циклы: 5
Состояния: 18
Адресация: непосредственная/косвенная регистровая
Флаги: нет

C (УСЛОВИЕ) (адр.) (УСЛОВНЫЙ ВЫЗОВ)
если (CCC), то ((SP)-1)←(PCH),
((SP)-2)←(PCL), (SP)←(SP)-2
(PC)←(байт 3) (байт 2)

Если указанное условие истинно, то выполняются действия, описанные в команде CALL, в противном случае последовательность исполнения команд не изменяется.

1	1	C	C	C	1	0	0
Младший байт адреса							
Старший байт адреса							

CNZ (адр.) C4
CZ (адр.) CC
CNC (адр.) D4
CC (адр.) DC
CPO (адр.) E4
CPE (адр.) EC
CP (адр.) F4
CM (адр.) FC

Циклы: 5
Состояния: 18
Адресация: непосредственная/косвенная регистровая
Флаги: нет

RET (ВОЗВРАТ)

(PCL)←((SP))
(PCH)←((SP)+ 1)
(SP)←(SP)+2

Содержимое ячейки памяти, адрес которой содержится в указателе СТЕКА SP, пересылается в 8 младших бит программного счетчика PC. Содержимое ячейки памяти, адрес которой на 1 больше содержимого указателя СТЕКА, пересылается в 8 старших бит программного счетчика. Содержимое указателя СТЕКА увеличивается на 2.

1	1	0	0	1	0	0	1

RET C9

Циклы: 3
Состояния: 10
Адресация: косвенная регистровая
Флаги: нет

R условие (УСЛОВНЫЙ ВОЗВРАТ)

если (CCC), то (PCL)←((SP)),
(PCH)←((SP)+ 1), (SP)←((SP)+2)

Если указанное условие истинно, то выполняются действия, описанные в команде RET, в противном случае последовательность исполнения команд не нарушается.

1	1	C	C	C	0	0	0

RNZ CO
RZ C8
RNC DO
RC D8
RPO EO
RPE E8
RP FO
RM F8

Циклы: 3
Состояния: 12
Адресация: косвенная регистровая
Флаги: нет

RST n (РЕСТАРТ)

((SP)-1) ← (PCH)
((SP)-2 ← (PCL)
(SP) ← ((SP)--2)
(PC)← 8* (NNN)

Старшие 8 бит адреса следующей команды пересылаются в ячейку памяти,

адрес которой на 1 меньше содержимого указателя СТЕКА. Младшие 8 бит адреса следующей команды пересылаются в ячейку памяти, адрес которой на 2 меньше содержимого указателя СТЕКА. Содержимое указателя СТЕКА уменьшается на 2. Управление передается команде, адрес которой равен коду NNN, умноженному на 8.

1	1	N	N	N	1	1	1

RST 0 C7
RST 1 CF
RST 2 D7
RST 3 DF
RST 4 E7
RST 5 EF
RST 6 F7
RST 7 FF

Циклы: 3
Состояния: 12
Адресация: косвенная регистровая
Флаги: нет

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Программный счетчик после рестарта.

PCHL (ЗАГРУЗКА СОДЕРЖИМОГО РЕГИСТРОВ H и L В ПРОГРАММНЫЙ СЧЕТЧИК)

(PCH) ← (H)
(PCL) ← (L)

Содержимое регистра H пересылается в 8 старших бит программного счетчика PC. Содержимое регистра L пересылается в 8 младших бит программного счетчика.

1	1	1	0	1	0	0	1

PCHL E9

Циклы: 1
Состояния: 6
Адресация: регистровая
Флаги: нет

КОМАНДЫ ВВОДА/ВЫВОДА, УПРАВЛЕНИЯ РАБОТЫ СО СТЕКОМ

Команды этой группы выполняют операции ввода/вывода, изменяют содержимое СТЕКА и состояние флагов.

Регистр флагов

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

X - безразлично

PUSH rp (ЗАСЫЛКА В СТЕК СОДЕРЖИМОГО ПАРЫ РЕГИСТРОВ)

((SP)-1) ← (rh)
((SP)-2) ← (rl)
(SP) ← (SP)-2

Содержимое старшего регистра пары регистров rp пересылается в ячейку памяти, адрес которой на 1 меньше содержимого указателя СТЕКА SP. Содержимое младшего регистра пары регистров rp пересылается в ячейку памяти, адрес которой на 2 меньше содержимого указателя СТЕКА. Содержимое указателя СТЕКА уменьшается на 2.

Примечание. Пара регистров rp=SP не может быть указана в команде.

1	1	R	P	0	1	0	1

PUSH B C5 (заслать пару регистров B и C)
PUSH D D5 (заслать пару регистров D и E)
PUSH H E5 (заслать пару регистров H и L)

Циклы: 3
Состояния: 12
Адресация: косвенная регистровая
Флаги: нет

PUSH PSW (ЗАСЛАТЬ В СТЕК СЛОВО СОСТОЯНИЯ ПРОГРАММЫ)

((SP)-1) ← (A)
$((SP)-2)_0$ ← (CY), $((SP)-2)_1$ ← X
$((SP)-2)_2$ ← (P), $((SP)-2_3$ ← X
$((SP)-2)_4$ ← (AC), $((SP)-2)_5$ ← X
$((SP)-2)_6$ ← (Z), $((SP)-2)_7$ ← (S)
(SP) ← (SP)-2

X - безразлично

Содержимое аккумулятора пересылается в ячейку памяти, адрес которой на 1 меньше содержимого указателя СТЕКА SP. Содержимое флагов объединяется в слово состояния, и это слово, пересылается в ячейку памяти, адрес которой на 2 меньше содержимого указателя СТЕКА. Содержимое указателя СТЕКА уменьшается на 2.

1	1	1	1	0	1	0	1

PUSH PSW F5

Циклы: 3
Состояния: 12
Адресация: косвенная регистровая
Флаги: нет

POP rp (СЧИТАТЬ ИЗ СТЕКА СОДЕРЖИМОЕ ПАРЫ РЕГИСТРОВ)

(rl) ← ((SP))
(rh) ← ((SP)+1)
(SP) ← (SP)+2

Содержимое ячейки памяти, адрес которой определяется содержимым указателя СТЕКА SP, пересылается в младший регистр пары регистров rp. Содержимое ячейки памяти, адрес которой на 1 больше содержимого указателя СТЕКА, пересылается в старший регистр пары регистров rp. Содержимое указателя СТЕКА увеличивается на 2.

Примечание. В команде не может быть указана пара регистров rp=SP.

1	1	R	P	0	0	1

POP B C1 (считать из СТЕКА пару регистров B и C)
POP D D1 (считать из СТЕКА пару регистров D и E)
POP H E1 (считать из СТЕКА пару регистров H и L)

Циклы: 3
Состояния: 10
Адресация: косвенная регистровая
Флаги: нет

POP PSW (СЧИТАТЬ ИЗ СТЕКА СЛОВО СОСТОЯНИЯ ПРОГРАММЫ)

((CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP)) +1)
(SP) ← (SP) +2

Содержимое ячейки памяти, адрес которой определяется содержимым указателя СТЕКА SP, используется для восстановления состояния флагов. Содержимое ячейки памяти, адрес которой на 1 больше содержимого указателя СТЕКА, переключается в аккумулятор. Содержимое указателя СТЕКА увеличивается на 2.

1	1	1	1	0	0	0	1

POP PSW F1

Циклы: 3
Состояния: 10
Адресация: косвенная регистровая
Флаги: Z, S, P, CY, AC

XTHL (ОБМЕН СОДЕРЖИМОГО ВЕРХУШКИ СТЕКА И СОДЕРЖИМОГО РЕГИСТРОВ H и L)

(L) ↔ ((SP))
(H) ↔ ((SP) +1)

Содержимое регистра L обменивается на содержимое ячейки памяти, адрес которой содержится в указателе СТЕКА sp. Содержимое регистра H обменивается на содержимое ячейки памяти, адрес которой на 1 больше содержимого указателя СТЕКА.

1	1	1	0	0	0	1	1

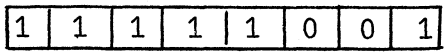
XTHL E3

Циклы: 5
Состояния: 16
Адресация: косвенная регистровая
Флаги: нет

SPHL (ПЕРЕСЫЛКА СОДЕРЖИМОГО РЕГИСТРОВ H И L В УКАЗАТЕЛЬ СТЕКА)

(SP) ↔ (H) (L)

Содержимое регистров H и L (16 бит) пересылается в указатель СТЕКА).



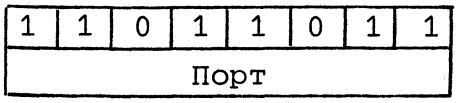
SPHL F9

Циклы: 1
Состояния: 6
Адресация: регистровая
Флаги: нет

IN порт (ВВОД)

/A/ ←— (Данные)

Данные, имеющиеся на восьми битах двунаправленной шины данных указанного порта, пересылаются в аккумулятор.



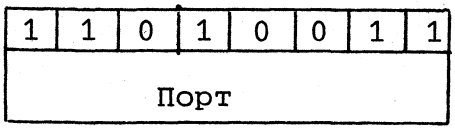
IN порт DB

Циклы: 3
Состояния: 10
Адресация: прямая
Флаги: нет

OUT порт (ВЫВОД)

(Данные) ← (A)

Содержимое аккумулятора помещается на двунаправленную шину данных для передачи в указанный порт.

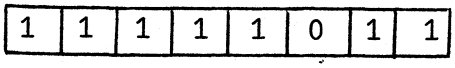


OUT порт D3

Циклы: 3
Состояния: 10
Адресация: прямая
Флаги: нет

EI (РАЗРЕШЕНИЕ ПРЕРЫВАНИЙ)

Система прерываний разрешается при исполнении следующей команды.



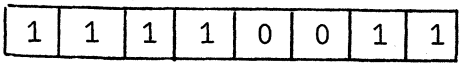
EI FB

Циклы: 1
Состояния: 4
Флаги: нет

Примечание. Прерывания не распознаются во время исполнения команды EI.

DI (ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЙ)

Система прерываний запрещается непосредственно при выполнении следующей за DI командой.



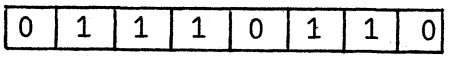
DI FB

Циклы: 1
Состояния: 4
Флаги: нет

Примечание. Прерывания не распознаются во время исполнения команды.

HLT (ОСТАНОВ)

В PC заносится адрес следующей команды. Процессор затем бездействует до прихода прерывания. Регистры и флаги не изменяют состояния.

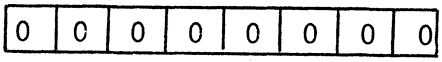


HLT 76

Циклы: 1
Состояния: 5
Флаги: нет

NOP (НЕТ ОПЕРАЦИИ)

В PC заносится адрес следующей команды, и МП переходит к ее обработке. Состояние флагов и регистров не изменяется.



NOP OO

Циклы: 1
Состояния: 4
Флаги: нет

ПРОГРАММА МОНИТОРА ДЛЯ "МИКРОЛАБ КР580ИК80"

```
                        ORG   0
0000    3E  92          MVI   A,92H          ;Управляющее слово для КР580ИК55
0002    D3  FB          OUT   OFBH           ;Программирование КР580ИК55
0004    C3              JMP   MONST
        3B  00
                        ORG   8H
0008    3E  92          MVI   A,92H          ;Управляющее слово для КР580ИК55
000A    D3  FB          OUT   OFBH           ;Программирование КР580ИК55
000C    C3              JMP   START
        55  00
                        ORG   10H
0010    C3              JMP   RST2
        D1  83
                        ORG   18H
0018    C3              JMP   RST3
        D4  83
                        ORG   20H
0020    C3              JMP   RST4
        D7  83
                        ORG   28H
0028    C3              JMP   RST5
        DA  83
                        ORG   30H
0030    C3              JMP   RST6
        DD  83
                        ORG   38H
0038    C3              JMP   BRENT
        51  01
```

Программа инициализации

```
003B    3E  FF  MONST:  MVI A,OFFH
003D    D3  FA          OUT OFAN        ;Шина С, инициализация 0 бита
003F    3E  8F          MVI A,8F
0041    D3  FA          OUT OFAN
0043    21              LXI H,DATA
0044    EC  83
0046    06  OC          MVI B,OC
0048    AF              XRA A
0049    77              MOV M,A
004A    23              INX H
004B    05              DCR B
004C    C2              JNZ S-3
004D    49  00
004F    21              LXI H,USESP
0050    C7  83
0052    22              SHLD SSAVE      ;Установка для СТЕКА пользователя
0053    E2  83
```

Начало монитора

```
0055    31              LXI SP,MONSP
        D1  83
```

```
0058    CD              CALL SEGCG      ;Преобразование сегментов
        CD  01
005B    CD              CALL KEVIN      ;Ввод с клавиш
        16  02
005E    47              MOV, B,A
005F    E6  10          ANI 1OH
0061    CA              JZ  DIGIT       ;Если 0, вводимые данные = 0___F
        84  00
0064    78              MOV A,B
0065    E6  OF          ANI OFH
0067    06  00          MVI B,O         ;B=0
0069    87              ADD A
006A    4F              MOV C,A
006B    21              BXI H,TABL
        74  00
006E    09              DAD B
006F    7E              MOV A,M
0070    23              INX H
0071    66              MOV H,M
0072    6F              MOV L,A
0073    E9              PCHL
0074    CC  00          TABL:DW  GOTO
0076    F9  01               DW  RESRG
0078    94  00               DW  ADSET
007A    B8  00               DW  ADDCX
007C    9D  00               DW  ADINX
007E    C2  00               DW  MEMW
0080    D5  00               DW  STAPE
0082    07  01               DW  LTAPE
0084    CD             DIGIT:CALL SHIFT ;Сдвиг регистра данных (4 бита)
        B5  01
0087    3A                   LDA DATA
        EC  83
008A    BO                   ORA B
008B    32                   STA DATA   ;Установление вводимых данных
        EC  83
008E    CD                   CALL RGDSP ;Высвечивание регистра данных и
        A1  01                          адреса
0091    C3                   JMP START
        55  00
```

Установка адреса

```
0094    2A          ADSET:LHLD DATA  ;HL = регистр данных
        EC  83
0097    22                SHLD ADRES ;Занос HL в регистр адреса
        EE  83
009A    C3                JMP ADINX+4;Чтение памяти и высвечивание адреса
        A1  00
```

Чтение памяти и приращение адреса

```
009D  2A          ADINX:LHLD ADRES       ;HL=регистр адреса
      EE  83
00A0  23          INX H                  ;Приращение адреса
```

```
00A1    CD          CALL MEMR               ;Чтение памяти
        AD  00
00A4    22          ADSTR:SHLD ADRES        ;Занос HL в регистр адреса
        EE  83
00A7    CD          CALL RGDSP              ;Высвечивание адреса и данных
        A1  01
00AA    C3          JMP START
        55  00
00AD    3A          MEMR:LDA DATA
        EC  83
00BO    32          STA DATA + 1            ;Сдвиг регистра данных
        ED  83
00B3    7E          MOV A,M                 ;Чтение памяти
00B4    32          CTA DATA                ;Данные____регистр данных
        EC  83
00B7    C9          RET
```

Чтение памяти и уменьшение адреса

```
00B8    2A          ADDCX:LHLD ADRES        ;HL=регистр адреса
        EE  83
00BB    2B              DCX H               ;Уменьшение адреса
00BC    CD              CALL MEMR           ;Чтение памяти
        AD  00
00BF    C3              JMP ADSTR
        A4  00
```

Запись в память

```
00C2    2A          MEMW:LHLD ADRES         ;HL=регистр адреса
        EE  83
00C5    3A              LDA DATA            ;A = регистр данных
        EC  83
00C8    77              MOV M,A             ;Запись данных
00C9    C3              JMP ADINX
        9D  00
```

Монитор программы управления пользователя

```
00CC    2A          GOTO:LHLD ADRES         ;HL=регистр адреса
        EE  83
00CF    22              SHLD PSAVE          ;(HL)_____(PC) область запоминания
        EO  83
00D2    C3              JMP RESRG           ;Восстановление регистра
        F9  01
```

Загрузка данных на ленту

```
00D5    OE  00  STAPE:MVI C,O               ;C = регистр контрольной суммы
00D7    2A              LHLD DATA           ;HL = конечный адрес
        EC  83
00DA    EB              XCHG                ;DK = конечный адрес
00DB    2A              LHLD ADRES          ;HL = начальный адрес
        EE  83
00DE    7C              MOV A,H
```

```
00DF  CD 41 01        CALL CKSMO      ;Вывод начального адреса [HI]
00E2  7D              MOV A,L
00E3  CD 41 01        CALL CKSMO      ;Вывод начального адреса [LO]
00E6  7A              MOV A,D
00E7  CD 41 01        CALL CKSMO      ;Вывод конечного адреса [HI]
00EA  7B              MOV A,E
00EB  CD 41 01        CALL CKSMO      ;Вывод конечного адреса [LO]
00EE  2B              DCX H
00EF  23        TAPE 1:INX H
00FO  7E              MOV A,M
00F1  CD 41 01        CALL CKSMO      ;Контрольная сумма и вывод данных
00F4  7D              MOV A,L
00F5  BB              CMP E           ;Сравнение начала и конца [LO]
00F6  C2 EF 00        JNZ TAPE 1
00F9  7C              MOV A,H
00FA  BA              CMP D           ;Сравнение начала и конца [HI]
00FB  C2 EF 00        JNZ TAPE 1
00FE  79              MOV A,C
00FF  2F              CMA
0100  3C              INR A
0101  CD 41 01        CALL CKSMO      ;Вывод контрольной суммы
0104  C5 55 00        JMP START       ;Конец ленты
```

Загрузка данных с ленты

```
0107  3E 01     LTAPE:MVI A,01H
0109  D3 FA           OUT OFAN        ;Запрещение к DMA
010B  OE 00           MVI C,O         ;C=регистр контрольной суммы
010D  CD 49 01        CALL CKSMI      ;Чтение данных и контрольная сумма
0110  67              MOV H,A         ;H=начальный адрес [HI]
0111  CD 49 01        CALL CKSMI      ;Чтение данных и контрольная сумма
0114  6F              MOV L,A         ;L=начальный адрес [LO]
0115  CD 49 01        CALL CKSMI      ;Чтение данных и контрольная сумма
0118  57              MOV D,A         ;D=конечный адрес [HI]
0119  CD 49 01     CALL CKSMI         ;Чтение данных и контрольная сумма
011C  5F           MOV E,A            ;E=конечный адрес [LO]
011D  22 EE 83     SHLD ADRES         ;Запоминание стартовых адресов
                                       в адресном регистре
0120  EB           XCHG
0121  22          SHLD DATA          ;Запоминание конечного адреса в
                                       регистре данных
      EC 83
0124  EB           XCHG
0125  2B           DCX H
```

```
0126    23      TAPE2:INX H
0127    CD                              ;Чтение данных и контрольная сумма
        49  01
012A    77          MOV M,A             ;Занос данных в память
012B    7D          MOV A,L
012C    BB          CMP E               ;Сравнение начала и конца [LO]
012D    C2          JNZ TAPE2
        26  01
0130    7C          MOV A,H
0131    BA          CMP D               ;Сравнение начала и конца [HI]
0132    C2          JNZ TAPE 2
        26  01
0135    CD          CALL CKSMI          ;Чтение данных и контрольная сумма
        49  01
0138    C2          JNZ ERROR           ;Если флаг 0-0___ошибка контроль-
                                         ной суммы
        CB  02
013B    CD          CALL RGDSP
        A1  01
013E    C3          JMP START           ;Конец загрузки данных
        55  00
0141    F5          CKSMO:PUSH PSW      ;Запоминание PSW
0142    81          ADD C               ;Контрольная сумма
0143    4F          MOV C,A
0144    F1          POP PSW             ;Восстановление PSW
0145    CD          CALL SRIOT          ;Вывод данных
        7C  02
0148    C9          RET
0149    CD          CKSMI:CALL SRIIN    ;Чтение данных
        AO  02
014C    47          MOV B,A             ;Запоминание ACC
014D    81          ADD C               ;Контрольная сумма
014E    4F          MOV C,A
014F    78          MOV A,B             ;Восстановление ACC
0150    C9          RET
```

Ввод прерывания

Прерывание и одношаговая работа

```
0151    E3          BRENT:XTHL          ;HL____PC (Запомнено)
0152    22              SHLD PSAVE      ;PC(LO)--83EO,PC(HI)____83El
        EO  83
0155    F5              PUSH PSW        ;Запоминание PSW
0156    21              LXI H,4H
        04  00
0159    39              DAD SP          ;HL----SP
015A    F1              POP PSW         ;Восстановление PSW
015B    22              SHLD SSAVE      ;SP(LO)--83E2,SP(HI)--83E3
        E2  83
015E    E1              POP H           ;Восстановление HL
015F    31              LXI SP,DATA
        EC  83
0162    F5              PUSH PSW        ;A--83EB,F--83EA запоминание
0163    C5              PUSH B          ;B--83E9,C--83E8 запоминание
0164    D5              PUSH D          ;D--83E7,E--83E6 запоминание
0165    E5              PUSH H          ;H--83E5,L--83E4 запоминание
0166    31              LXI SP,MONSP    ;Инициализация SP
        D1  83
```

```
0169  3A F2 83        LDA BRKCT
016C  A7               ANA A              ;Счетчик прерывания = 0 или нет
016D  CA 8B 01         JZ BSTOP           ;Если 0, то 1 шаг
0170  2A FO 83         LHLD BRKAD
0173  EB               XCHG               ;DE = указатель прерывания
0174  2A EO 83         LHLD PSAVE         ;HL = (PC)
0177  7D               MOV A,L
0178  BB               CMP E              ;PC(LO) =BP(LO)
0179  C2 85 01         JNZ NOBRK          ;Если = 0, то нет прерывания
017C  7C               MOV A,H
017D  BA               CMP D              ;PC(HI)-BP(HI)
017E  C2 85 01         JNZ NOBRK          ;Если ≠ 0, то нет прерывания
0181  21 F2 83         LXI H,BRKCT
0184  35               DCR M              ;Уменьшение счетчика прерывания
0185  CD 91 01   NOBRK: CALL ADDSP        ;Высвечивание (PSW) и (PC)
0188  C3 F9 01         JMP RESRG          ;Восстановление регистра и возврат
018B  CD 91 01   BSTOP: CALL ADDSP        ;Высвечивание (PSW) и (PC)
018E  C3 55 00         JMP START
0191  2A EA 83   ADDSP: LHLD FSAVE        ;HL= (PSW)
0194  22 EC 83         SHLD DATA          ;(PSW)___Регистр данных
0197  2A EO 83         LHLD PSAVE         ;HL= (PC)
019A  22 EE 83         SHLD ADDRES        ;(PC)___Регистр адреса
019   CD A1 01         CALL RGDSP         ;Высвечивание регистра адреса и
                                           данных
01AO  C9               RET
```

ПОДПРОГРАММЫ

Высвечивание регистра адреса и данных

```
01A1  21 EF 83   RGDSP:LXI H,ADRES+1      ;HL=буферный адрес
01A4  11 F4 83         LXI D,DISP         ;Высвечивание данных буферного
                                           адреса
01A7  06 04            MVI B,4            ;Установление счетчика
01A9  7E               MOV A,M
01AA  12               STAX D             ;Высвечивание запомненных данных
01AB  2B               DCX H
01AC  13               INX D
01AD  05               DCR B
```

```
01AE    C2              JNZ S-5
        A9   01
01B1    CD              CALL SEGGG          ;Счетчик сегментов
        CO   01
01B4    C9              RET
```

Сдвиг регистра данных (4 бита)

```
01B5    2A      SHIFT:LHLD DATA
        EC   83
01B8    29              DAD H
01B9    29              DAD H
01BA    29              DAD H
01BB    29              DAD H
01BC    22              SHLD DATA
        EC   83
01BF    C9              RET
```

Программа счетчика сегментов

```
01CO    21      SEGGG:LXI H,DISP        ;HL=высвечивание адреса данных
        F4   83
01C3    11              LXI D,DIG       ;DE=сегмент буфера адреса
        F8   83
01C6    01              LXI B,SEGD      ;BC=сегмент буфера адреса
        E9   01
01C9    7E              MOV A,M
01CA    23              INX H
01CB    E5              PUSH H
01CC    F5              PUSH PSW
01CD    E6   FO         ANI OFOH        ;Маска "FO"
01CF    OF              RRC
01DO    OF              RRC
01D1    OF              RRC
01D2    OF              RRC             ;Сдвиг вправо 4 бита
01D3    26   00         MVI H,O
01D5    6F              MOV L,A
01D6    09              DAD B
01D7    7E              MOV A,M         ;A=сегмент данных
01D8    12              STAX D          ;Запоминание данных сегмента
01D9    13              INX D
01DA    F1              POP PSW
01DB    E6   0F         ANI OFH         ;Маска "OF"
01DD    26   00         MVI H,O
01DF    6F              MOV L,A
01EO    09              DAD B
01E1    7E              MOV A,M         ;A=сегмент данных
01E2    12              STAX D          ;Запоминание данных сегмента
01E3    E1              POP H
01E4    1C              INR E
01E5    C2              JNZ SEGCG+9
        C9   01
01E8    C9              RET
```

Сегментные данные

```
01E9   5C 06  SEGD:DB   5CH,06H,5EH,4FH,66H,6DH
       5B 4F
       66 6D
01EF   7D 27       DB   7DH,27H,7FH,6FH,77H,7CH,39H
       7F 6F
       77 7C
01F6   39          DB   5EH,79H,71H
       5E
       79 71
```

Восстановление регистра

```
01F9   2A      RESRG:LHLD SSAVE
       E2 83
01FC   F9            SPHL                ;Восстановление SP
01FD   2A            LHLD PSAVE
       EO 83
0200   E5            PUSH H              ;PC запоминание в СТЕКЕ пользо-
                                          вателя
0201   2A            LHLD LSAVE
       E4 83
0204   E5            PUSH H              ;HL запоминание в СТЕКЕ пользо-
                                          вателя
0205   2A            LHLD FSAVE
       EA 83
0208   E5            PUSH H              ;PSW запоминание в СТЕКЕ пользо-
                                          вателя
0209   2A            LHLD CSAVE
       E8 83
020C   4D            MOV C,L
020D   44            MOV B,H             ;BC восстановлено
020E   2A            LHLD ESAVE
       E6 83
0211   EB            XCHG                ;DE восстановлено
0212   F1            POP PSW             ;PSW восстановлено

0213   E1      POP H                     ;HL восстановлено
0214   FB      EI                        ;Разрешение прерывания
0215   C9      RET                       ;Восстановление PC и перевод в
                                          программу пользователя
```

Ввод с клавиши

ACC=ввод данных

```
0216   CD      KEYIN:CALL RESET          ;Сброс
       69 02
0219   16 00   KEY 2:MVI D,O
021B   42            MOV B,D
021C   3E 9F        MVI A,09FH
```

```
021E    D3    FA          OUT OFAH
0220    CD                CALL WAT              ;Задержка
        62    02
0223    DB    F8          IN OF8H               ;Сканирование клавиатуры
                                                 0___7
0225    EE    FF          XRI OFFH              ;Дополнение
0227    C2                JNZ KEY1
        50    02
022A    06    08          MVI B,8
022C    3E    AF          MVI A,OAFH
022E    D3    FA          OUT OFAN
0230    CD                CALL WAT              ;Задержка
        62    02
0233    DB    F8          IN OF8H               ;Сканирование клавиатуры 8___F
0235    EE    FF          XRI OFFH              ;Дополнение
0237    C2                JNZ KEYI
        50    02
023A    06    10          MVI B,1OH
023C    3E    CF          MVI A,OCFH
023E    D3    FA          OUT OFAH
0240    CD                CALL WAT              ;Задержка
        62    02
0243    DB    F8          IN OF8H               ;Сканирование клавиатуры
0245    EE    FF          XRI OFFH              ;Дополнение
0247    C2                JHZ KEYI
        50    02
024A    32                STA KFLAG
        F3    83
024D    C3                JMP KEY2
        19    02
0250    OF        KEY1:   RRC
0251    DA                JC S+3
        58    02
0254    14                INR D
0255    C3                JMP S-3
        50    02
0258    7A                MOV A,D
0259    BO                ORA B                 ;Изменение данных
025A    47                MOV B,A
025B    3E    FF          MVI A,FF
025D    32                STA KFLAG
        F3    83
0260    78                MOV A,B
0261    C9                RET
0262    1E    FF  WAT:    MVI E,FF              ;Длительность задержки
0264    1D                DCR E
0265    C2                JNZ S-1
        64 02
0268    C9                RET
0269    3E    92  RESET:  MVI,A 92
026B    D3    FB          OUT OFBH
026D    3E    FF          MVI A,FF
026F    D3    FA          OUT OFAH
0271    DB    F8          IN OF8H
0273    EE    FF          XRI OFFH
0275    C2                JNZ S-4
        6D    02
0278    C9                RET
```

Подпрограмма вывода

```
027C  D5       SRIOT:PUSH D          ;Запоминание DE регистра
027D  C5             PUSH B          ;Запоминание BC регистра
027E  06  08         NVI B,8         ;Установка счетчика битов
0280  4F             MOV C,A         ;C_____ACC
0281  AF             XRA A           ;Установка стартового бита
0282  D3  FA         OUT OFAH        ;Вывод стартового бита
0284  CD             CALL D2         ;Ожидание
      EA  02
0287  79       SRI01:MOV  A,C        ;ACC____данные
0288  E6  7F         ANI 7FH         ;Маска M.S.B.
028A  D3  FA         OUT OFAH        ;Вывод L.S.B.
028C  79             MOV A,C
028D  1F             RAR             ;Сдвиг данных
028E  4F             MOV C,A
028F  CD             CALL D2         ;Ожидание
      EA  02
0292  05             DCR B           ;Уменьшение счетчика бита
0293  C2             JNZ SRI01
      87  02
0296  3E  01         MVI A,1         ;Установка конечного бита
0298  D3  FA         OUT OFAH        ;Вывод конечного бита
029A  CD             CALL D3         ;Ожидание
      EF  02
029D  C1             POP B           ;Выборка BC регистров из СТЕКА
029E  D1             POP D           ;Выборка DE регистров из СТЕКА
029F  C9             RET
```

Подпрограмма ввода

```
02A0  D5       SRIIN:PUSH D          ;Запоминание DE регистров
02A1  C5             PUSH B          ;Запоминание BC регистров
02A2  01             LXI B,8O OH     ;Инициализация
      00  08
02A5  DB  F9   SRII1:IN OF9H         ;Получение входных данных
02A7  1F             ;Проверка L.S.B.
02A8  DA             JC SRII1        ;Если 0____SRII1
      A5  02
02AB  CD             CALL D1         ;Ожидание
      DD  02
02AE  DB  F9         IN OF9H         ;Получение входных данных
02BO  1F             RAR             ;Проверка L.S.B.
02B1  DA             JC SRII1        ;Если 1____SRII1
      A5  02
02B4  CD             CALL D2         ;Ожидание
      EA  02
02B7  DB  F9         IN OF9H         ;Получение входных данных
02B9  E6  01         ANI 1           ;Маска L.S.B.
02BB  81             ADD C           ;Сложить C и ACC
02BC  OF             RRC             ;Сдвиг данных
02BD  4F             MOV C,A         ;Данные ACC___C
02BE  05             DCR B           ;Уменьшение в регистре
02BF  C2             JNZ S-11        ;Перехват_02B4
      B4  02
```

```
02C2    CD              CALL D2             ;Ожидание
        EA   02
02C5    CD              CALL D2             ;Ожидание
        EA   02
02C8    C1              POP B               ;Выборка BC регистров из СТЕКА
02C9    D1              POP D               ;Выборка DE регистров из СТЕКА
02CA    C9              RET
02CB    21        ERROR:LXI H,DIG           ;HL =сегмент данных буферного
        F8   83                              адреса

02CE    36   79         MVI M,79H
02DO    23               INX H
02D1    36   80         MVI M,8OH
02D3    2C               INR L
02D4    C2               JNJ S-3           ;Запись сообщения о дефекте
        D1   C2
02D7    31               LXI SP,MONSP      ;Инициализация SP
        D1   83
02DA    C3               JMP START+10
        5B   00
```

Ожидание и таймер диалога

```
02DD    16   24   D1:     MVI D,24H         ;Ожидание 4.5112 мс
02DF    1E   OC           MVI E,OCH
02E1    1D                DCR E
02E2    C2                JNZ S-1
        E1   02
02E5    15                DCR D
02E6    C2                JNZ S-7
        DF   02
02E9    C9                RET
02EA    16   48       D2:MVI D, 48H        ;Ожидание 9.0176 мс
02EC    C3                JMP D1+2
        DF   02
02EF    16   D8       D3:MVI D, OD8H       ;Ожидание 27.0176 мс
02F1    C3                JMP D1+2
        DF   02
```

Таблица адресов

```
LSAVE:EOU       83E4H
HSAVE:EOU       83E5H
ESAVE:EOU       83E6H
DSAVE:EOU       83E7H
CSAVE:EOU       83E8H
```

СОДЕРЖАНИЕ

Введение. Микропроцессоры и микро-ЭВМ

Урок 1. Основные элементы микропроцессорной системы 3
Урок 2. Системы счисления 8
Урок 3. Основы матобеспечения 11
Урок 4. Использование "Микролаб КР580ИК80" 17
Урок 5. Основные принципы программирования 21
Урок 6. Структура и функционирование микропроцессора КР580ИК80 27

Аппаратные средства

Урок 7. Основные принципы построения аппаратной части 29
Урок 8. Дешифрация адреса 34
Урок 9. Запоминающие устройства и периферийные устройства 37
Урок 10. Сигналы управления в микропроцессорной системе 41

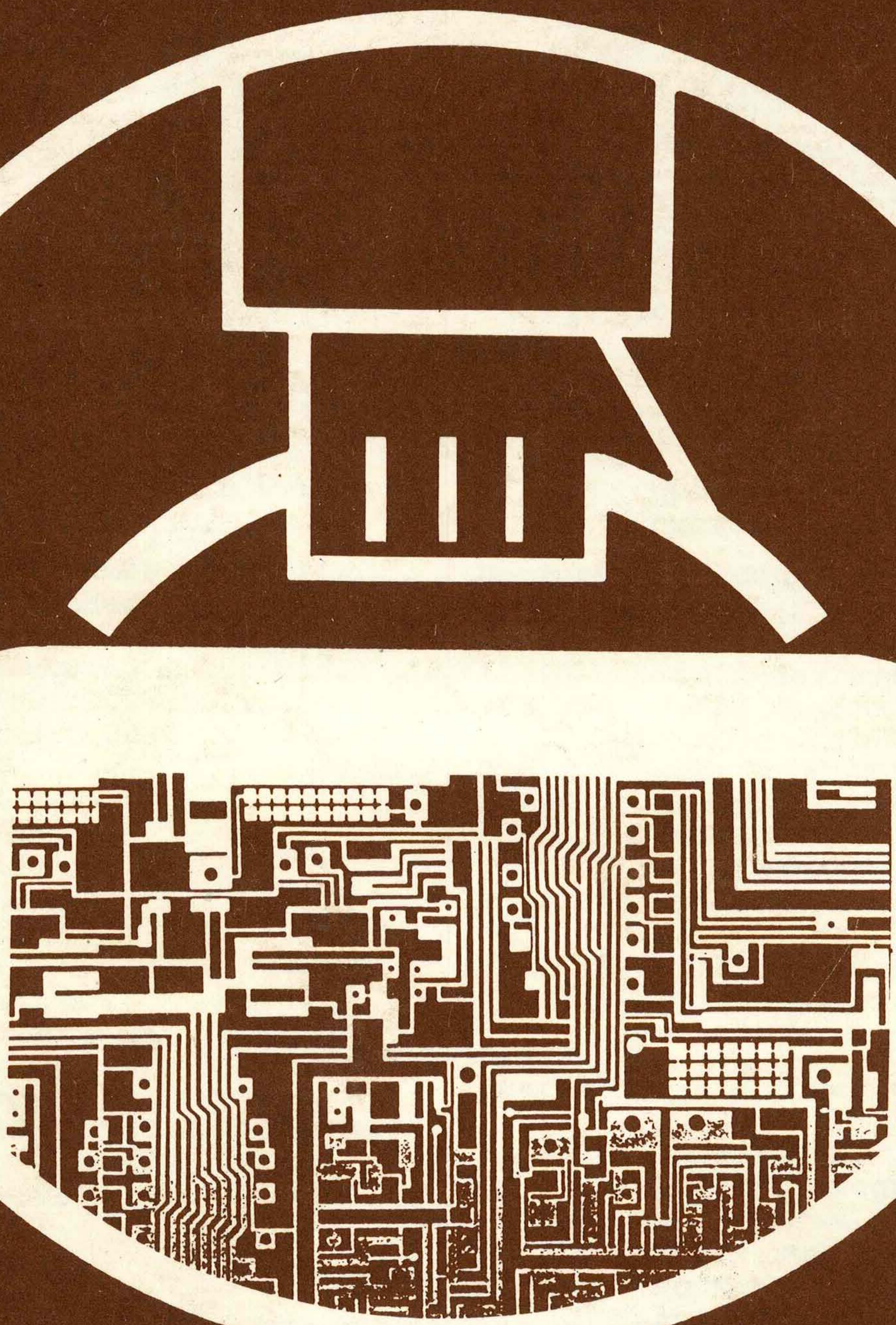
Программные средства

Урок 11. Регистры и точки прерывания 45
Урок 12. Набор команд микропроцессора КР580ИК80 50
Урок 13. Методы разработки программного обеспечения 57
Урок 14. Клавиатура и индикация 62
Урок 15. Некоторые представления чисел и алгоритмы 67

Диагоностика и поиск неисправности

Урок 16. Портативные диагностические устройства. Поиск неисправностей в микропроцессорных системах 71
Урок 17. Сигнатурные анализаторы 73
Урок 18. Логические анализаторы 77
Урок 19. Диагностика микропроцессорных систем 80

Приложение 93



Внешторгиздат.Изд.№ 19064эс.
1 вклейка.
ВТИ.Зак. 3111